This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n: 761999

EasyTV: Easing the access of Europeans with disabilities to converging media and content.

D3.2 Enriched multilingual ontology with signs in different languages preliminary version

EasyTV Project

H2020. ICT-19-2017 Media and content convergence. – IA Innovation action.

Grant Agreement n°: 761999

Start date of project: 1 Oct. 2017

Duration: 30 months

Document. ref.: Deliverable 3.2

Disclaimer

The EasyTV Consortium consists of the following partners:

	Partner Name	Short name	Country
1	Universidad Politécnica de Madrid	UPM	ES
2	Engineering Ingegneria Informatica S.P.A.	ENG	IT
3	Centre for Research and Technology Hellas/Information Technologies Institute	CERTH	GR
4	Mediavoice SRL	MV	IT
5	Universitat Autònoma Barcelona	UAB	ES
6	Corporació Catalana de Mitjans Audiovisuals SA	CCMA	ES
7	ARX.NET SA	ARX	GR
8	Fundación Confederación Nacional Sordos España para la supresión de barreras de comunicación	FCNSE	ES

PROGRAMME NAME:	H2020. ICT-19-2017 Media and content convergence. convergence. – IA Innovation action
PROJECT NUMBER:	761999
PROJECT TITLE:	EASYTV
RESPONSIBLE UNIT:	UPM
INVOLVED UNITS:	UPM, CERTH
DOCUMENT NUMBER:	D3.2
DOCUMENT TITLE:	Enriched multilingual ontology with signs in different languages preliminary version
WORK-PACKAGE:	3
DELIVERABLE TYPE:	R
CONTRACTUAL DATE OF DELIVERY:	31-09-2018
LAST UPDATE:	20-09-2018
DISTRIBUTION LEVEL:	PU

Distribution level:

PU = *Public,*

RE = *Restricted to a group of the specified Consortium,*

PP = *Restricted to other program participants (including Commission Services),*

CO = *Confidential, only for members of the LASIE Consortium (including the Commission Services)*

Document History

VERSION	DATE	STATUS	AUTHORS, REVIEWER	DESCRIPTION
v.0.1	02/08/2018	Draft	María Poveda-Villalón (UPM)	Table of Contents definition and document structure
v.0.2	20/08/2018	Draft	María Poveda-Villalón (UPM) Pablo Calleja-Ibañez (UPM) Elena Montiel-Ponsoda (UPM)	Added first version of sections 2, 4, 5 and 6
v.0.3	23/08/2018	Draft	María Poveda-Villalón (UPM) Julia Bosque-Gil (UPM) Elena Montiel-Ponsoda (UPM)	Added first version of sections 1 and draft for section 3.
v.0.4	24/08/2018	Draft	Pablo Calleja-Ibañez (UPM)	Added first draft of section 8.
v.0.5	27/08/2018	Draft	María Poveda-Villalón (UPM) Julia Bosque-Gil (UPM) Elena Montiel-Ponsoda (UPM)	Added first version of section 3.
v.0.6	29/08/2018	Draft	María Poveda-Villalón (UPM) Julia Bosque-Gil (UPM) Elena Montiel-Ponsoda (UPM)	Completed first version of section 3.
v.0.7	30/08/2018	Draft	Pablo Calleja-Ibañez (UPM)	Added first version of section 8.
v.0.8	10/09/2018	Draft	Kosmas Dimitropoulos (CERTH) Kiriakos Stefanidis (CERTH)	Added section 7.
v.0.9	12/09/2018	Draft	María Poveda-Villalón (UPM) Julia Bosque-Gil (UPM) Elena Montiel-Ponsoda (UPM)	Modifications over section 5.

v.0.10	17/09/2018	Draft	Pilar Orero (UAB)	review
v.0.11	18/09/2018	Draft	Stavros Skourtis (ARX)	review
v.0.12	19/09/2018	Draft	María Poveda-Villalón (UPM) Elena Montiel-Ponsoda (UPM) Pablo Calleja-Ibañez (UPM)	Updates according review
v.0.13	20/09/2018	Draft	Kosmas Dimitropoulos (CERTH) Kiriakos Stefanidis (CERTH)	Updates according review
v.1.0	20/09/2018	First version	María Poveda-Villalón (UPM)	Consolidated changes

Definitions, Acronyms and Abbreviations

ACRONYMS / ABBREVIATIONS	DESCRIPTION
API	Application Programming Interface
HTML	Hyper Text Markup Language
OWL	Web Ontology Language
PoS Tagger	Part of Speech Tagger
RDF	Resource Description Framework
RDF(S)	RDF Schema
SPARQL	SPARQL Protocol and RDF Query Language
UML	Unified Modeling Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium

Table of Contents

List of Figures

List of Tables

EXECUTIVE SUMMARY

The present document is the deliverable "D3.2.1 Enriched multilingual ontology with signs in different languages preliminary version" of the EasyTV project,[1] funded as an Innovation Action by the European Commission Directorate-General for Research and Innovation (DG RTD), under its Horizon 2020 Research and Innovation Programme (H2020).

This deliverable gives an overview and documents the EasyTV ontology as of September 2018. The documentation covers, among others, the following topics:

- The **EasyTV ontology** is presented in this deliverable including the description of the current implementation and resources available, and the modelling decisions taken up to the time of writing this document.
- The description of relevant semantic models about linguistic information to be reused in the development of the EasyTV ontology, and the semantic repository data generation including an EasyTV ontology **example** of use.
- The description of existing API and tools for natural language processing relevant for the semantic data annotation process (to be carried out by the EasyTV annotation module) that will use for the EasyTV ontology to populate the semantic repository.
- **The Relationship** with other **modules** of the project where the multilingual ontology is used, more precisely within the crowdsourcing platform. The connection between the EasyTV annotation module with other modules as the Sign Language capturing system is also explained.

In addition, the document provides an overview of the methodological guidelines and recommendable infrastructure to be used during the ontology development process as an annex. The deliverable includes some conclusions and future lines of work.

[1] http://easytvproject.eu/

1. INTRODUCTION

The EasyTV crowdsourcing platform relies on ontologies (i.e., semantic data models) that will be exploited in order to annotate Sign Language videos with the aim of easing the translation between videos in different Sign Languages. In computer science, ontologies are defined as "formal, explicit specifications of a shared conceptualization" [16]. The EasyTV ontology will be formal in the sense of following Description Logics and being implemented in the W3C Web Ontology Language standard OWL.[2] The EasyTV ontology will be used within the crowdsourcing platform as reference model to annotate Sign Language videos with linguistic information. The annotations will stablish links between the videos and a multilingual knowledge base and will be exploited to facilitate the matching between Sign Language videos previously stored and annotated. This document will describe in detail the process followed to build such ontology and its current model.

The scope of this deliverable is not limited to the description of the resulting ontology. The processes followed and the infrastructure deployed during the ontology development are also explained. In addition, some existing ontologies and natural language processing tools are reviewed as well as the exchange data format to be taken as input of the semantic annotation process.

The remainder of this deliverable is structured as follows:

- **Section 2** provides an introduction to the semantic technologies that are planned to be used within EasyTV crowdsourcing platform in order to link Sign Language videos to ease finding potential translations.
- **Section 3** provides an overview of the main existing ontologies related to the EasyTV ontology scope.
- **Section 4** describes the infrastructure deployed during the ontology development process as well as the current implementation of the EasyTV ontology.
- **Section 5** describes main linguistic libraries and APIs available to be used during the Sign Language video processing and annotation processes.
- **Section 6** focuses on the video exchange format and the transcription information between the video capturing and the semantic annotation models.
- **Section 7** provides some conclusions and future lines of work.

For the sake of understandability of this document, readers not familiar with the concept of ontologies in computer science and the Web, might find the basis for this topic in the "Ontology Development 101: A Guide to Creating Your First Ontology".[3] Also, the methodological guidelines intended to be followed during the EasyTV ontology development are sketched in Annex 1. In addition, the EasyTV deliverable "D 5.2.1 Mid-term report on the set up and implementation of the EasyTV crowdsourcing Sign Language platform and repository" can provide background knowledge to readers about how the EasyTV ontology and the semantic technologies are planned to be used within the crowdsourcing platform.

[2] https://www.w3.org/TR/owl-ref/

[3] http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html

2. SEMANTIC WEB TECHNOLOGIES

The semantic web is an extension of the Web where the meaning (semantics) of the information and services are defined according to agreed data models, also called ontologies. The World Wide Web consortium[4] provides a number of technologies to make this notion of semantic web a reality, including languages for data representation (Resource Description Framework (RDF)) and ontology implementation (as Resource Description Framework Schema (RDF(S))and Web Ontology Language (OWL)) as well as for querying such data and models (SPARQL Protocol and RDF Query Language (SPARQL)). Such technologies have been developed in the context of the "W3C Data Activity"[5] and will be further explained in this section.

The volume of semantic data published over the Web has experienced a huge growth in the last decade, mainly boosted by the Linked Data[6] initiative. The Linked Data concept started with the idea of using the Web to "connect data" and it has been transforming the current web, where the connected items are documents, into a global database in which data is connected to other data by means of ontologies.

There is a fundamental question about knowledge base structure when publishing Linked Data. On the one hand developers should define data models (ontology) and, on the other hand, one should create the data (resources) that will be described by means of the ontology, and will be published on the web as Linked Data. To benefit from the semantic web advantages and features, the data annotation and publication should be driven by a validated, correct and robust ontology, developed following methodological guidelines.

In computer science, the term ontology is used to refer to a "formal, explicit specification of a shared conceptualization" [16]. Conceptualization refers to an abstract model that allows describing something relevant in the world, for which we normally use concepts, properties and constraints on their application (e.g., the Unified Modelling Language (UML) class diagrams that many software developers use, the entity-relationship models used to organise a database, or any drawing that one makes in a whiteboard to start organising an information model for a system development). All those entities in the abstract model need to be described explicitly aiming at covering as much as possible of the world phenomenon that we are trying to represent. For example, if we are talking about different types of persons or organisations, we should include the different categories of persons and organisations that are involved in our model of the world, as well as the relationships and constraints that hold among them. Being formal refers to the fact that the ontology should be machine-readable (that is, available in some language (e.g., RDF-S or OWL) that can be easily processed). And finally, and most importantly, "shared" reflects the notion that an ontology captures consensual knowledge, that is, it is not private of an individual, but accepted by a group.

In the following paragraphs, different technologies used for the development and publication of Linked Data are described. As shown in Figure 1, one of the base technologies is the data model RDF, used for data exchange. On top of RDF, the ontology implementation languages RDF-S and OWL are built. Finally, we present the SPARQL language which has been developed as query language for RDF data.

[4] https://www.w3.org/

[5] http://www.w3.org/2013/data/

[6] https://www.w3.org/standards/semanticweb/data

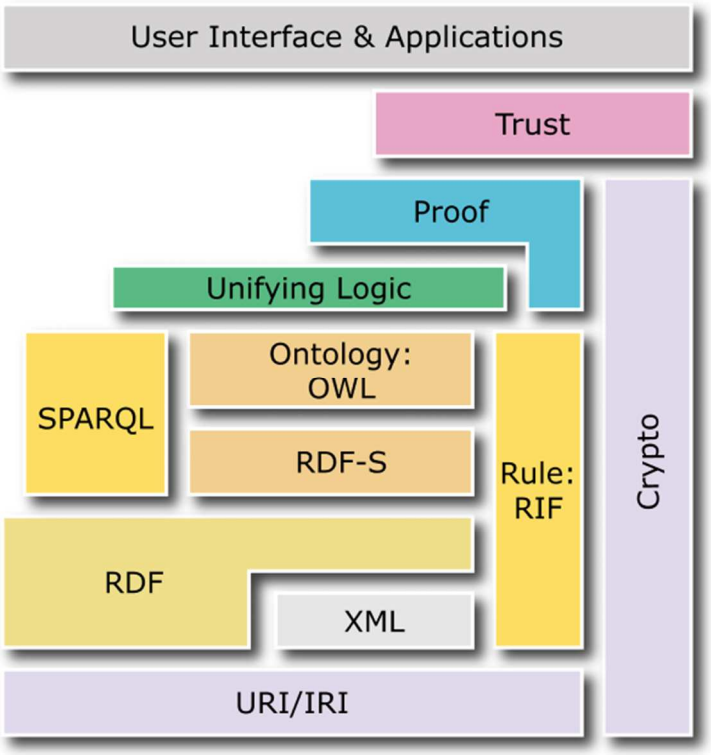

Figure 1. Semantic Web technology stack (W3C)

RDF[7] **(Resource Description Framework)** is an standard data model to describe resources on the web. The basic structure of this data model is called "triple" and it is composed of three elements, namely: "subject", "predicate", and "object". This data model represents the basis for the linked data developments, that is, the technology used to represent data, classifications, and their relations. The publication of linked data in RDF could be carried out in different ways, for example, by means of a SPARQL endpoint or publishing RDF dumps among others.

OWL[8] **(Web Ontology Language)** is an ontology implementation language designed to represent shared ontologies on the web based on RDF. This language extends the capabilities of other ontology languages as RDF-S by means of incorporating a new vocabulary with attached formal semantics that incorporates inference features. OWL allows creating classes, hierarchies and properties, as well as RDF-S, but also creating local axioms constraints for classes. Such constraints involve existential and universal quantifiers. OWL also provides means to add characteristics to the properties as functional, transitivity, symmetry, etc.

SPARQL[9] **(SPARQL Protocol and RDF Query Language)** is a query language for RDF data established as official recommendation of the W3C. This language allows for mandatory or optional pattern matching in order to query the database. In addition, SPARQL allows for the application of constraints over the queries by stating the graph(s) over which the query should be checked. The results of the SPARQL queries could be a set of values or RDF graphs.

[7] http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140225/

[8] http://www.w3.org/TR/owl-ref/

[9] http://www.w3.org/TR/rdf-sparql-query/

In addition to this technologies, in the context of Linked Data some recommendations have been made in order to guide the development and publication of data. First, we should mention the Linked Data principles[10] described by Tim Berners-Lee, the web inventor, taken literarily:

1. *Use URIs[11] as names for things*
2. *Use HTTP URIs so that people can look up those names.*
3. *When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)*
4. *Include links to other URIs, so that they can discover more things.*

Finally, apart from the basic linked data publication principles above-mentioned, a 5-star rating schema is provided stablishing best practices for data publication on the web. In particular, these practices emphasize the need not only to publish data using standard technologies, but also to do so in an open fashion. In this way we moved from the idea of linked data the notion of open and linked data (Linked Open Data). The 5-star scheme describes the following levels at which the data should (taken literarily):

★ *Available on the web (whatever format) but with an open licence, to be Open Data*

★★ *Available as machine-readable structured data (e.g. excel instead of image scan of a table)*

★★★ *as (2) plus non-proprietary format (e.g. CSV instead of excel)*

★★★★ *All the above plus, Use open standards from W3C (RDF and SPARQL) to identify things, so that people can point at your stuff*

★★★★★ *All the above, plus: Link your data to other people's data to provide context*

Each star-level reach by a published dataset is determined in an incremental way. That is, in order to have a dataset of 2 stars (available as structured data) first it should comply with all the previous levels, in this case it should be available under an open license.

In summary, this chapter has reviewed the main semantic technologies that will support the extension of the multilingual knowledge base with Sign Language videos described in task T3.2 More precisely, we can state that RDF is the data model to be used to store semantic annotations for Sign Language videos. Such annotations are driven by the EasyTV ontology which is being developed following the OWL ontology implementation language. The RDF containing the annotations will be loaded in a triple store that will provide querying capabilities by means of the SPARQL query language.

[10] https://www.w3.org/DesignIssues/LinkedData.html

[11] URI: Uniform Resource Identifier

3. RELEVANT SEMANTIC MODELS FOR EASYTV

EasyTV crowdsourcing platform aims to facilitate the Sign Language video translation by stablishing links between videos and concepts from a multilingual knowledge base, namely BabelNet. Such links and the navigation through the knowledge base concepts in different language would be exploited to match videos and their potential translations or videos with similar meaning. BabelNet is one of the main resources from the Linguistic Linked Open Data (LLOD) [4] landscape. This LLOD represents an initiative about publishing data for linguistics and natural language processing using the linked data principles presented in Section 2.

In the remainder of this section main ontologies and resources to represent linguistic information are presented. More precisely, only those ontologies and models that are reused in the EasyTV ontology are described. For further information about available ontologies that have been developed for representing linguistic information by means of semantic technologies, we suggest interested readers to explore the Linguistic Linked Open Data working group[12] and to read the review of linguistic data models published by Bosque-Gil and colleagues [3].

3.1. SKOS

The Simple Knowledge Organization System (SKOS) vocabulary is provided as a W3C recommendation from August, 2009, broadly adopted by the semantic web community. This model is designed to link and share knowledge organization systems, being formalized in OWL and defining mainly concept schemas and concepts. The SKOS vocabulary is defined under its URI http://www.w3.org/2004/02/skos/core.

Knowledge Organization systems are different types of schemas to organize information in order to ease the knowledge management. These types of schemas include classifications, thesauri, glossaries, dictionaries, etc. In order to contribute to the semantic web and facilitate the translation of these schemas into RDF, the SKOS vocabulary was created. SKOS in an ontology[13] developed in OWL that provides a model for representing the basic structure and content of KOS systems.

A basic use of SKOS allows identifying conceptual resources (concepts) assigning them URIs, labelling them with literals of one or several languages, documenting them with different types of notes, relating them to each other through informal hierarchical structures or associative networks, and aggregating them to concept schemes. As SKOS in RDF-oriented, it allows the creation and publication of concepts on the web, as well as linking them with data in this same way and even integrate them into other concept schemes.

Figure 2 shown main concepts and relationships defined in the SKOS vocabulary. As it can be observed, the main classes are skos:ConceptSchema, used to represent the KOS systems, as an entity, in RDF and the class skos:Concept, which is used to represent the concepts collected in the given KOS system schema.

It is important to mention that to represent a KOS system it is necessary to create a URI that uniquely identifies it. For example, when transforming the UNESCO thesaurus, the individual that represents such UNESCO thesaurus would be an instance of the class skos:ConceptSchema. In addition, a URI must be created for each concept collected by the thesaurus at hand, such URIs are defined as

[12] http://linguistic-lod.org

[13] SKOS – Simple Knowledge Organization System http://www.w3.org/TR/skos-primer/

instances of the class skos:Concept. By means of the relationship skos:inSchema links are stablished between each of the concepts defined in a KOS system and the system they belong to. In this way, given a concept it can be retrieved the knowledge organization system in which such a concept is defined. In addition, through the relationship skos:hasTopConcept and its inverse skos:topConceptOf, the concepts of the first level of the classification or KOS system are identified.

Between different instances of the class skos:Concept serveral relationships could hold as it is shown in Figure 2. The main relationships are skos:narrower and its inverse skos:broader that allow to establish hyponymy relations (indicate which concepts are more specific) and hypernym (indicate which concepts are more general) between concepts respectively.

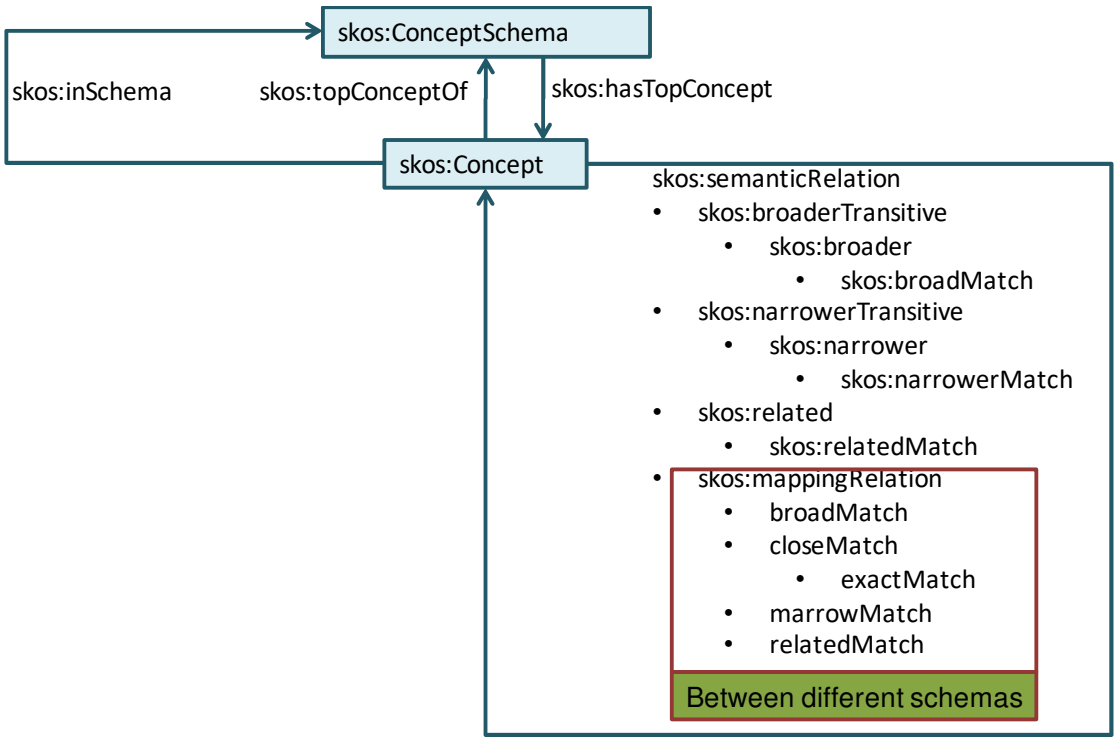

Figure 2. SKOS vocabulary overview

As also shown in Figure 2, SKOS defined a number of relations to link concepts defined in different classifications or knowledge organization systems. Such relations are materialized in the SKOS ontology by the following properties: skos:mappingRelation, skos:broadMatch, skos:closeMatch, skos:exactMatch, skos:narrowMatch and skos:relatedMatch.

3.2. Lemon

The Lexicon Model for Ontologies (lemon) [10], developed in the framework of the Monnet[14] European project, proposes a formal way for modelling lexicon and machine-readable dictionaries. The lemon model is based on the combination, review and extension of previous models such as LingInfo [2], LexOnto [6], LIR [11] and the W3C model SKOS.[15] This model has been design taking

[14] https://cordis.europa.eu/project/rcn/93713_en.html

[15] https://www.w3.org/TR/skos-primer/

into account the following challenges (taken literarily from the model website[16]):

- *RDF-native form to enable leverage of existing Semantic Web technologies (SPARQL, OWL, RIF etc.).*
- *Linguistically sound structure based on LMF to enable conversion to existing offline formats.*
- *Separation of the lexicon and ontology layers, to ensure compatibility with existing OWL models.*
- *Linking to data categories, in order to allow for arbitrarily complex linguistic description.*
- *A small model using the principle of least power - the less expressive the language, the more reusable the data.*

In addition, the lemon model follows a modular approach and it comprises five modules, namely: (1) core module for the representation of grammatical, (basic) morphological and semantic information module; (2) lexical and terminological variation module; (3) phrase structure module; (4) syntactic frames module; and (5) morphological variation module.

Figure 3 shows an overview of the lemon core module. According with the idea of keeping the conceptual layer (the ontology) separated from the lexical layer, the class lemon:LexicalSense (shown in Figure 3) is designed as the bridge between a lexical entry (represented by the class lemon:Lexicalentry) and its meaning, that would be represented by the a concept from the ontology at hand (represented by a ellipse in Figure 3). As it can be observed, a lexical entry could be further classified as Word, Phrase or Part. Also, a lexical entry can be linked to its different forms by means of the property form or any of its specializations, lemon:canonicalForm, lemon:abstractForm or lemon:otherForm.

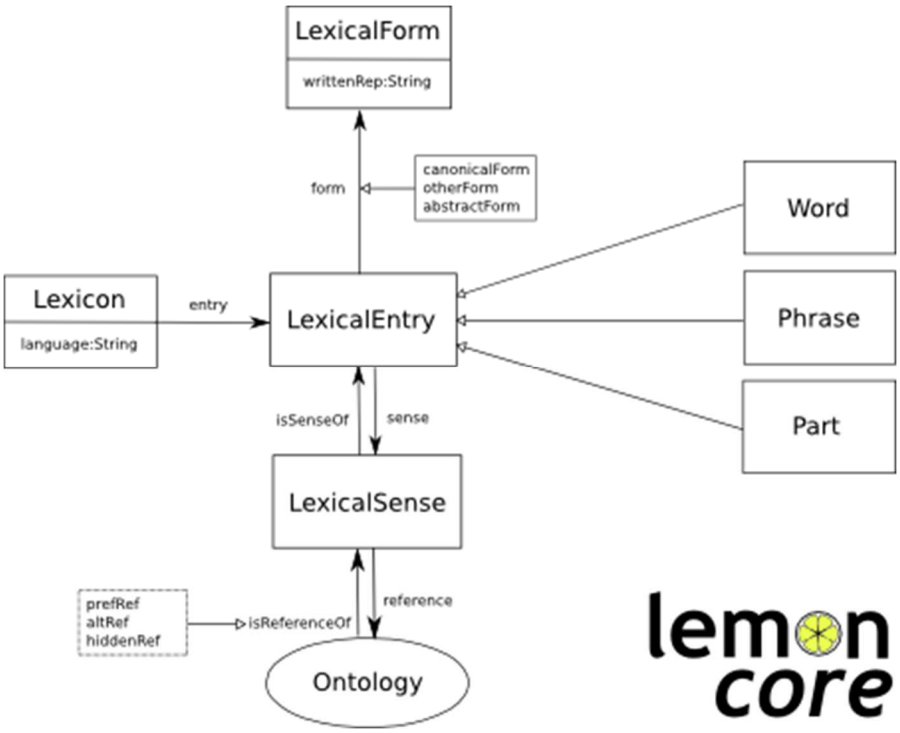

Figure 3. lemon core module diagram (taken from https://lemon-model.net/)

Even though lemon was devised for describing lexical information of ontologies as existing

[16] https://lemon-model.net/index.php

annotation system were limited to simple tagging, it became popular to represent lexica as Linked Data. In this sense, lemon is one of the models being used to convert lexica and other linguistic resources to Linked Data and has become a de facto standard to represent and interchange lexical data in the Semantic Web.

3.3. OntoLex

One of the W3C Ontology Lexicon (OntoLex) community group[17] main goal was to provide models to represent lexica and machine readable dictionaries relative to ontologies. This goal included: (1) the representation of lexical entries containing information about how ontology elements (classes, properties, individuals etc.) are realized in multiple languages and (2) the representation of linguistic information (syntactic, morphological, semantic and pragmatic) that constrains the usage of the entry.

This work was carried out taking Lexicon Model for Ontologies (lemon) [10] as input to be reviewed, extended and formally modularized by opening it to the community. The resulting model is officially named lemon[18] but also known as OntoLex or lemon-ontolex.

OntoLex is divided into five modules, each one implemented in a separate ontology. The core module (URI: http://www.w3.org/ns/lemon/ontolex#) describes lexical entries, their forms and their mapping to the denoted meaning in an ontology. An overview of this module is shown in Figure 4.[19] As it can be observed, while the main conceptualization for lexical entries, senses and forms is maintained from lemon model, OntoLex includes the representation of lexical concepts to represent mental abstraction, concepts or units of thought that can be lexicalized by a collection of senses, and concept sets that represent the collection of lexical concepts. This addition is implemented by means of extending the SKOS classes skos:Concept and skos:ConceptSchema with the ontolex:LexicalConcept and ontolex:ConceptSet, respectively.

The rest of the modules are organized as follows:

- synsem (URI: http://www.w3.org/ns/lemon/synsem#) is dedicated to represent the syntax-semantics interface
- vartrans (URI: http://www.w3.org/ns/lemon/vartrans#) is used to represent variations and translations
- decomp (http://www.w3.org/ns/lemon/decomp#) is oriented to the representation of the internal structure of an entry
- lime (http://www.w3.org/ns/lemon/lime#) is designed to represent linguistic metadata

[17] https://www.w3.org/community/ontolex/

[18] https://www.w3.org/2016/05/ontolex/

[19] This diagram is not provided oficially by the Ontology Lexicon community group, instead it has been created taking as input the actual ontolex OWL code. It should be mentioned that the diagram represents the axioms included in the code which might include some mistakes regarding the intended conceptualization as reported in https://thepetiteontologist.wordpress.com/2018/03/25/brief-analysis-of-ontolex/

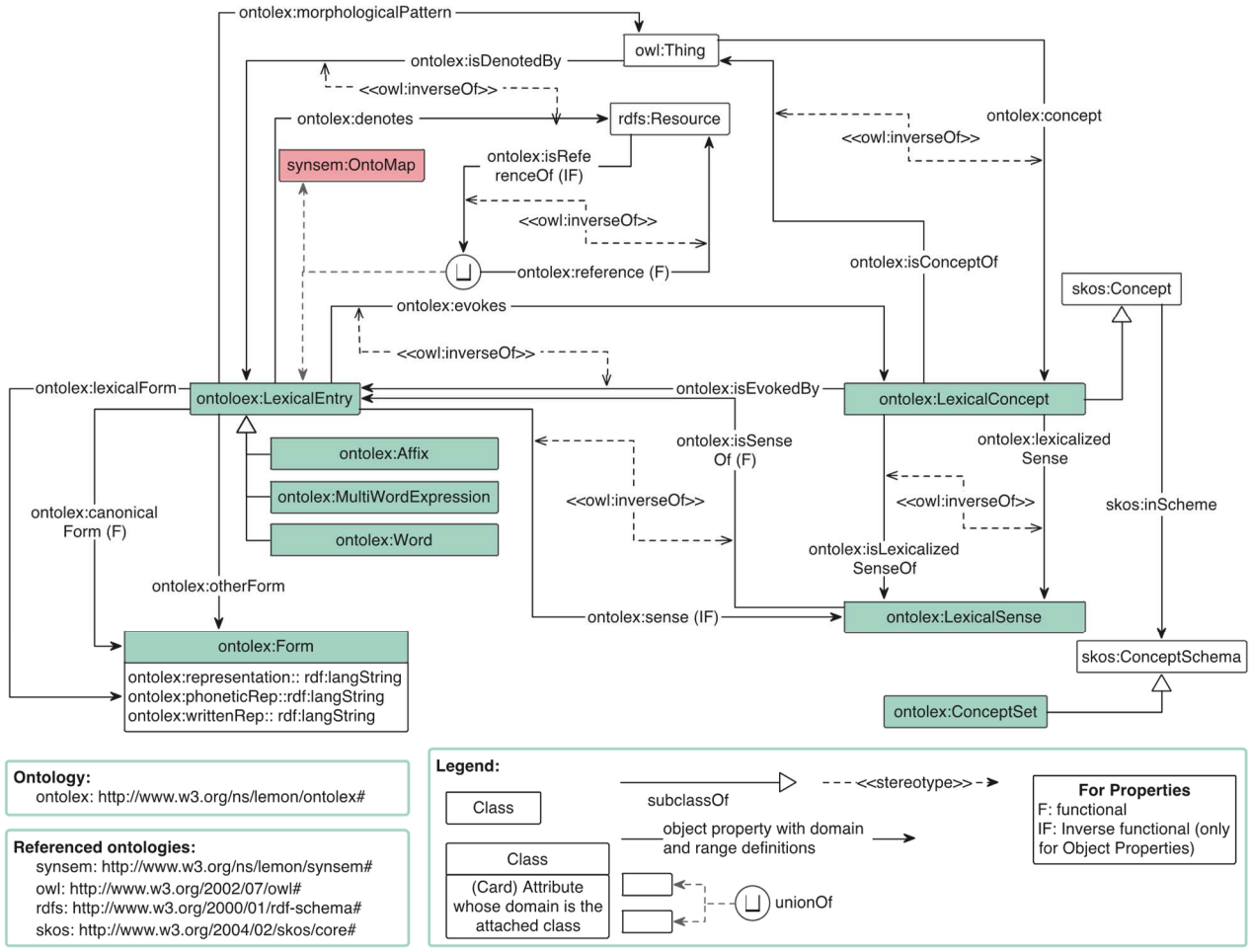

Figure 4. OntoLex core module overview

3.4. LexInfo

The LexInfo [5] ontology was developed in the context of the Monnet Project.[20] This model provides a linguistic data category registry by extending the lemon model and based on LingInfo [2] and LexOnto [6]. LexInfo is a lexicon model that tackles the limitations of SKOS, RDF and RDFS in encoding linguistic information associated to the elements of an ontology, and it is grounded on the separation of the conceptual and linguistic layers.

The second version of LexInfo is an extensive ontology of types, values and properties derived partially from ISOcat, and currently its elements capture information from the morphosyntactic, syntactic, syntactic-semantic, semantic and pragmatic levels of linguistic description.

Figure 5 shows a graphical representation of an excerpt of the LexInfo ontology. The figure include some lemon classes that are extended by LexInfo. It should be mentioned, that only the LexInfo elements more relevant for the EasyTV ontology are included in Figure 5. As it can be observed, LexInfo extends the lemon:Word class to include a hierarchy of part of speech categories, by means of the classes lexinfo:Adjective, lexinfo:Adverb, lexinfo:Noun, etc. among others. A subset of the

[20] https://lexinfo.net/

morphosyntactic properties definedin LexInfo are also depicted in Figure 5. The class lexinfo:MorphosyntacticProperty extends the lemon:PropertyValue concept in order to gather the hierarchy of morpohsintactic properties which are represented by the classes lexinfo:Gender, lexinfo:Number, lexinfo:Person, etc. The hierarchy is also represented in the relationships linking instances with morphosyntactic properties as the relationship lexinfo:morphosyntacticProperty is extended with one subproperty for each class in the morphosyntantic hierarchy, for example lexinfo:gender, lexinfo:number or lexinfo:person among others. Finally, it should be mentioned that LexInfo provides a set of individuals to represent the possible values of the morphosyntactic properties. For example, the instances lexinfo:firstPerson, lexinfo:secondPerson and lexinfo:thirdPerson represent the possible values for the property lexinfo:person.

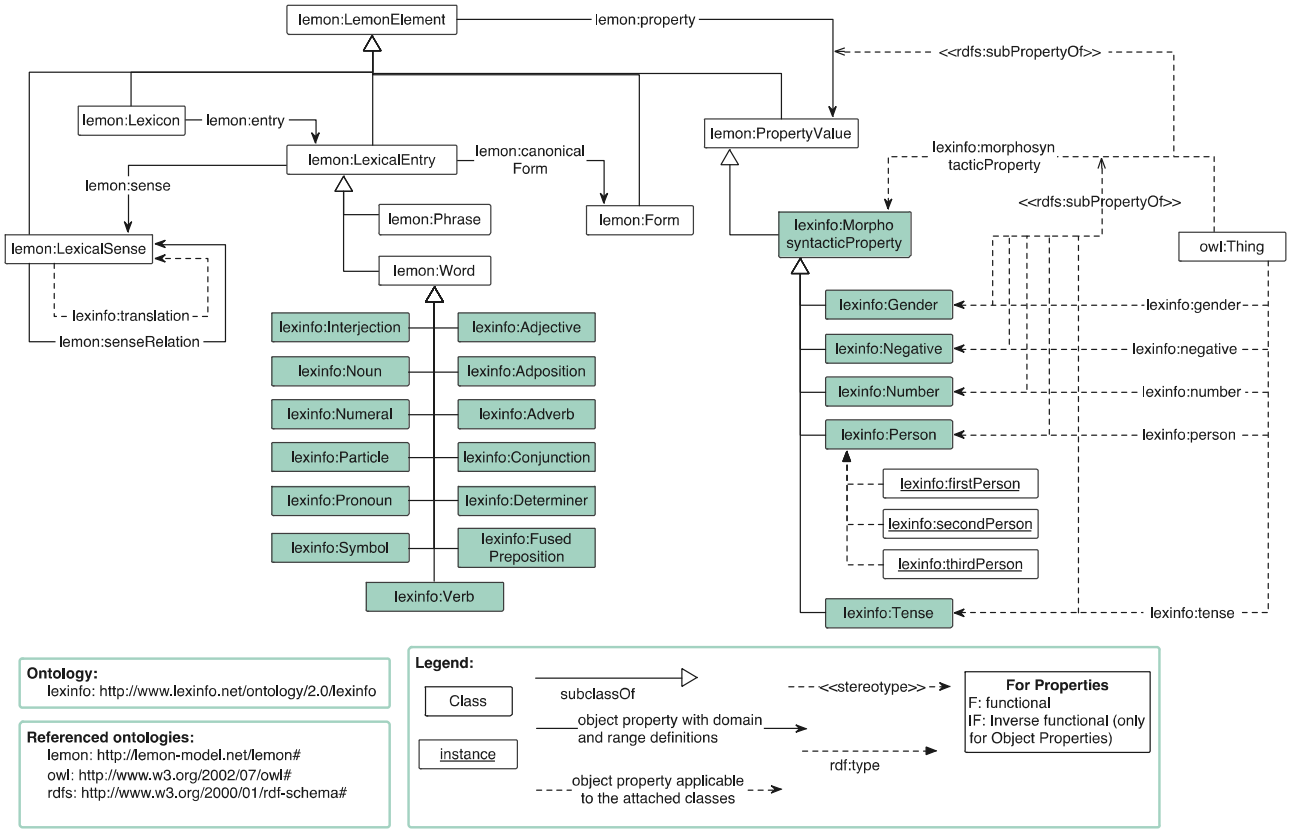

Figure 5. Excerpt of LexInfo model

3.5. BabelNet

BabelNet [12] is a multilingual encyclopedic dictionary, with lexicographic and encyclopedic coverage of terms, and a semantic network that connect concepts and named entities. The resulting semantic network is built by integrating different resources as Wikipedia, WordNet, Open Multilingual Wordnet, Wikidata, Wiktionary and OmegaWiki among others. BabelNet includes 16 million entries and each Babel synset represents a given meaning and contains all the synonyms expressing such meaning in different languages. In its version 4.0 BabelNet includes information in 284 languages. BabelNet provides access to its service[21] to navigate the semantic network both by API[22] and by a SPARQL

[21] https://babelnet.org/

[22] https://babelnet.org/guide

endpoint.[23] It should be mentioned that BabelNet APIs are provided under a non-commercial license.

The RDF data provided through the BabelNet SPARQL endpoint is modelled according to the BabelNet model[24] which is depicted in Figure 6. As it can be observed, the BabelNet is mostly based on the original lemon model and SKOS.

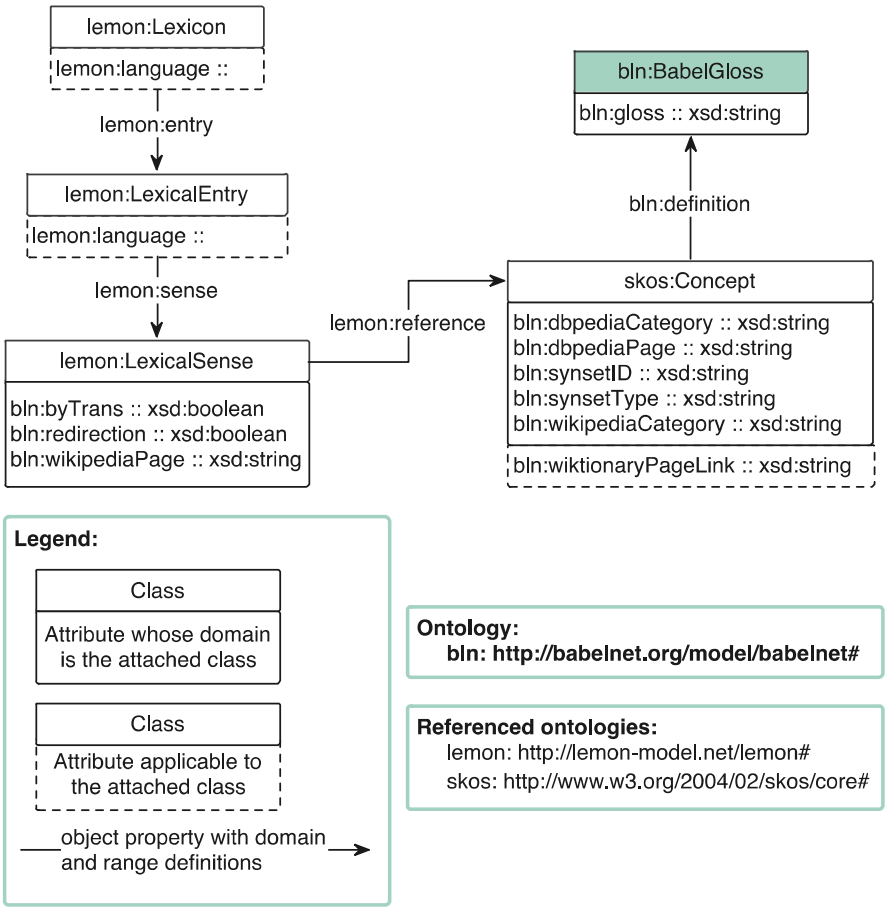

Figure 6. Overview of BabelNet model defined in https://babelnet.org/model/babelnet#

It should be clarified that Figure 6 depicts the BabelNet model explicitly declared in the ontology retrieved by the URI https://babelnet.org/model/babelnet#. However, the BabelNet RDF dataset makes use of other properties and classes not explicitly referenced from the mentioned ontology. In order to help understanding the model used in the BabelNet RDF dataset, we have explored the RDF data provided in the BabelNet SPARQL endpoint and come up with a graphical representation of the actual model underlying such data which is shown in Figure 7.

[23] https://babelnet.org/sparql/

[24] https://babelnet.org/model/babelnet#

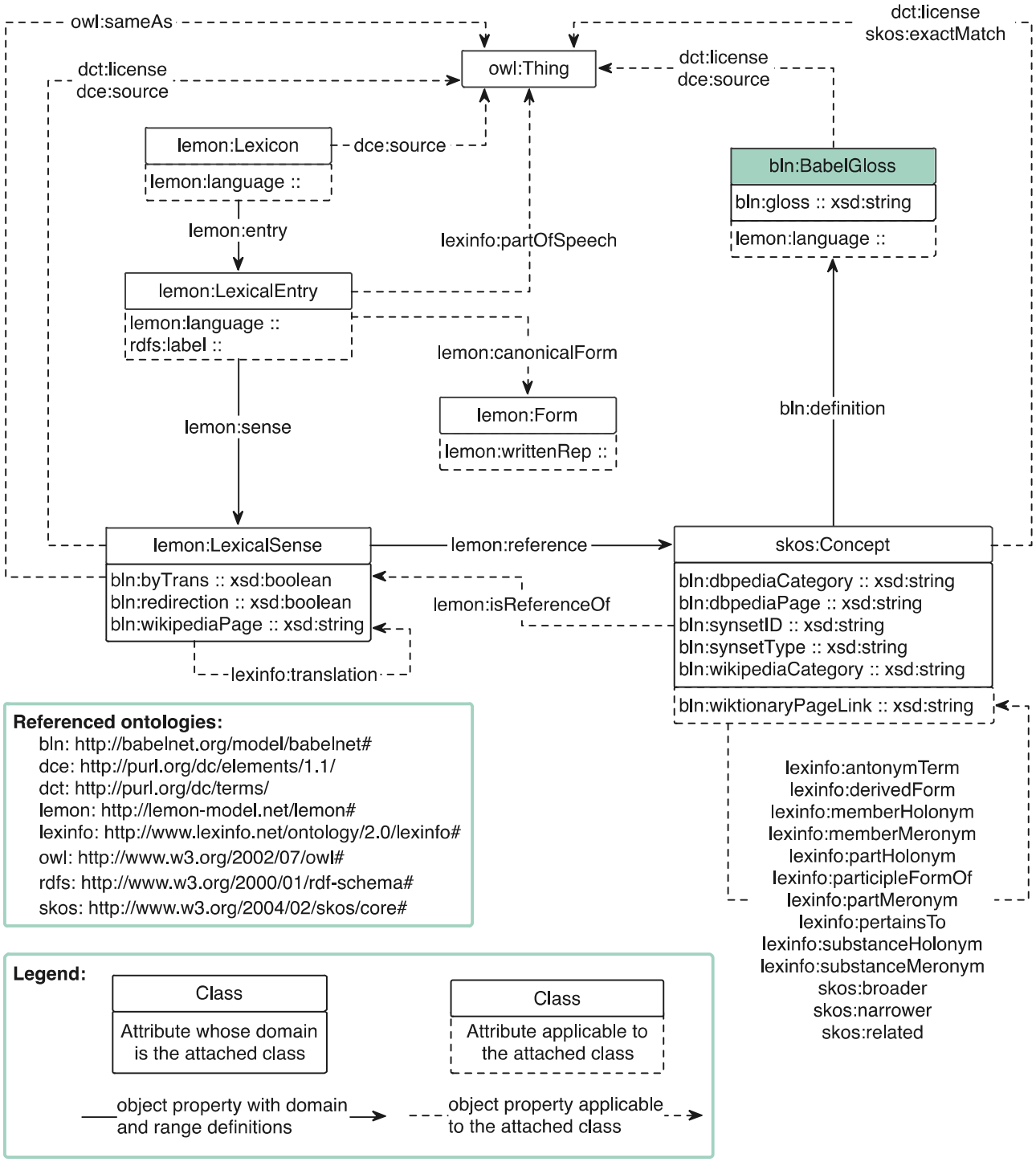

Figure 7. Underlying model used in BabelNet RDF dataset

Even though BabelNet comprises a huge amount of information, its conversion to RDF mainly involves two core concepts namely Babel senses and Babel synsets. A Babel synset is a collection of equivalent senses. While lemon provides the means of representing lexical information about the Babel senses, to represent Babel synsets the skos:Concept class is used in the BabelNet model. In addition, some properties from LexInfo are used to annotate some of the semantic relations between Babel synsets, more precisely lexinfo:translation. Along the BabelNet RDF dataset Babel sense

lemmas are encoded as lemon:LexicalEntry. Each lemon:LexicalEntry is annotated with a rdfs:label for each language and its part of speech, by means of the property lexinfo:partOfSpeech. Each entry is also linked to the corresponding lemon:Lexicon depending on their language.

Following the notion of semantics by reference modelled by lemon, the meanings of the lexical entries are indicated by Babel synsets (represented by skos:Concept) which are linked from the lexical senses (lemon:LexicalSense) by means of the property lemon:reference.

BabelNet includes provenance information for senses, for example the Wikipedia Page (bln:wikipediaPage) or, when relevant, the way it was obtained, via automatic translation (bln:byTrans) or thanks to a Wikipedia redirection page (bln:redirection or the source of the entity (dce:source).[25]

Finally, other semantic relations are included among Babel synsets taken from skos (e.g. skos:broader, skos:narrower or skos:related) and also taken from LexInfo (lexinfo:partHolonym, lexinfo:partMeronym, lexinfo:memberMeronym, etc.)

[25] “dce” represent the prefix for the URI http://purl.org/dc/elements/1.1/

4. EASYTV SEMANTIC MODEL PRELIMINAR VERSION

This section describes the ontology development infrastructure set up to support all the activities involved in the ontology development process and the preliminary version of the ontology being developed.

4.1. Infrastructure for the EasyTV ontology development

In order to support the **ontology requirements specification** phase, the ontology developers use a Google Spreadsheets to gather and store the functional requirements. This spreadsheet is associated to the EasyTV ontology, published online and openly accessible[26]. An overview of the current state of the functional ontological requirements is shown in Figure 8.

Identifier (domain+id)	Sprint	Competency Question / Natural language statement	Answer	Status (Proposed, Accepted, Rejected, Deprecated)	Superseded by	Comments	Extracted from (provenance)
slv1	1	A sign language video is expressed in a given language		A			
slv2	1	A sign language video represents a phrase in written language grammar		A			
slv3	1	A sign language video represents a phrase in sign language grammar		A			
slv4	1	A sign can be expressed by a video		A			
slv5	1	A sign language video might be compossed by frames		A			
slv6	1	A lexical entry has forms		A			
slv7		A sign or group of signs is the sign language form of a linguistic expression		A			
slv8		A phrase is a lexical entry		R			http://lemon-model.net/lemon#
slv9		Lexical forms and signs have morphosyntactic properties		A			http://lemon-model.net/lemon#
slv10		Gender is a morphosyntactic property		A			http://lemon-model.net/lemon#
slv11		Negation is a morphosyntactic property		A			http://lemon-model.net/lemon#
slv12		Number is a morphosyntactic property		A			http://lemon-model.net/lemon#
slv13		Person is a morphosyntactic property		A			http://lemon-model.net/lemon#
slv14		A phrase can by decomponsed by components		A			http://lemon-model.net/lemon#
slv15	1	A lexical entry has a lexical sense		A			Babelnet
slv16	1	A lexical sense reference an idea or concept		A			Babelnet

Figure 8. Excerpt of the current ontological functional requirements for the EasyTV model

Regarding the ontology implementation phase, different tools are usually managed by the ontology development team to carry out the edition, storage and evaluation activities.

One of the available **ontology editors** is Protégé,[27] developed at the Stanford Medical Informatics (SMI) of Stanford University. This desktop application, which is distributed as open source, is composed by a core that is the ontology editor that can be expanded by the available extensions that provide greater functionality to the development environment. These extensions or plug-ins can have different functionalities such as import/export of ontologies, visualization, reasoning, etc. Protégé allows the creation, visualization and manipulation of ontologies in various representation formats. In the context of the EasyTV Project, Protégé is used by the ontology development team to generate the ontology code.

As solution for **ontology storage**, the EasyTV ontology is stored in a GitHub repository[28] which includes:

- A folder with the implementation of the ontology.
- A folder with the ontology modelling diagrams.

[26] http://bit.ly/easyTVreqs

[27] http://protege.stanford.edu/

[28] https://github.com/mariapoveda/easytv-onto

- A folder with the documentation of the ontology.

This repository is planned to be used also for issue tracking and gathering ontology releases historical information.

The ontology developers may use the GitHub or Git proposed workflow to develop the ontology, which can be summarized in the next steps:

1. Create a new branch from the master branch.
2. Add changes to the ontology and commit them. Each commit has to be associated with a commit message, which is a description explaining why a particular change was made.
3. Open a pull request to start a discussion over the changes.
4. If the pull request is approved, merge the new branch into the master branch.

The development team uses OnToology[29] to generate the **documentation** and to **evaluate** the ontology. OnToology is a web application that allows the integration of different ontological engineering activities while managing the ontology code and associated resourced in GitHub repositories. OnToology, which integrates Widoco,[30] OOPS! [14][31] and AR2DTool,[32] generates automatically a folder in the GitHub repository which includes all the resources: diagrams, documentation and evaluation report.

The structure of the folders generated by OnToology for each ontology contained in the GitHub repository is represented in Figure 9:

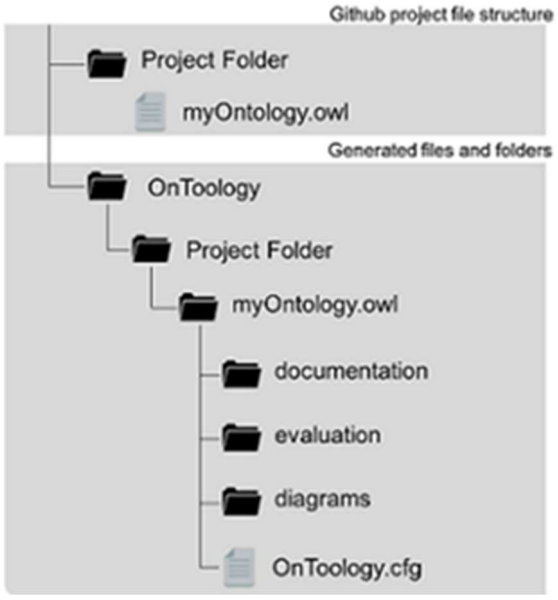

Figure 9. OnToology folder structure.

The documentation of the ontology is generated using Widoco, which provides a description of the classes that must be completed by the domain experts. The diagrams of the ontology are generated

[29] http://ontoology.linkeddata.es/
[30] https://github.com/dgarijo/Widoco/
[31] http://oops.linkeddata.es/
[32] https://github.com/idafensp/ar2dtool

using AR2DTool, which includes both taxonomy and entity relationship diagrams. The evaluation is provided by OOPS!, which detects the most common pitfalls that appear when developing ontologies. All the resources generated by OnToology are updated whenever there is a change in the ontology file.

To support the **ontology publication** phase, the ontology developers publish the ontology online to be accessible to anyone under its URI. The ontology development team publishes the releases of the ontologies to make the ontology code and its documentation accessible to all the users.

To support the **maintenance** of the ontology, the ontology developers use an **issue tracker**[33] which manages and maintain the list of issues identified by the domain experts and ontology developers.

All the changes and improvements over the ontology need to be agreed by all the ontology development team. The GitHub issue tracker[34] is used to discuss improvements and issues about the domain being modelled. If domain experts or ontology developers want to add new concepts to the ontology they have to create a new issue in the GitHub issue tracker, which will be used to start a discussion about the approval of the proposal.

GitHub issues will be marked as “open” and “closed”. The issues can also have an assignee which is the person that is responsible for moving the issue forward. Figure 10 shows the list of the opened issues associated to the EasyTV ontology.

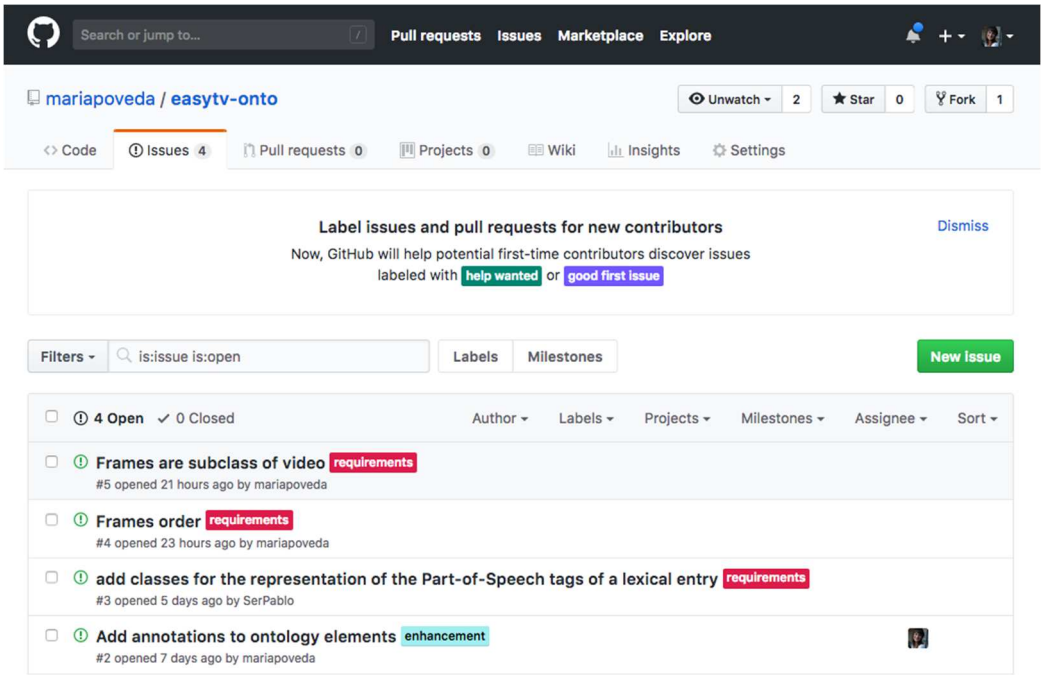

Figure 10. GitHub issues related to the EasyTV ontology.

The ontology developers should label each issue according to its topic or status in order to organize the different types of issues. GitHub comes with a few labels by default: “bug”, “duplicate”, “enhancement”, “invalid”, “question”, and “won't fix”. Those issues that represent new requirements

[33] https://github.com/mariapoveda/easytv-onto/issues

[34] https://help.github.com/articles/about-issues/

for the ontology will be labelled by the developers as a “requirement” issue. In addition, ontology developers can create new labels if they think they can improve issue management. The ontology developers are also responsible for closing the issues created that have already been addressed.

The new requirements which are proposed in a GitHub issue should be added to the ORSD associated to the ontology when the ontology development team approves them. The requirement is considered approved when the ontology developers tag the issue as “requirement” and add a comment associating an identifier and a competency question.

4.2. Overview of the current EasyTV model development

4.2.1. Ontological requirements

Following the steps proposed in 0 a first version of the ontological requirements to be fulfilled by the EasyTV ontology have been identified. The Ontology Requirement Specification Document fields are detailed as follows:

- **Purpose**: To describe concepts, relations and attributes to annotate Sign Language videos with linguistic information. The final goal is to link with multilingual semantic resources in order to facilitate translations between annotated videos.
- **Scope**: The scope of the ontology is limited to the linguistic information about the natural language text associated to the videos in which a sign or group of sign is represented. Therefore, the representation of the sign themselves are out of scope.
- **Ontological requirement**:
    - **Non-Functional requirements**:
        - The ontology should be documented at least in English.
        - The ontology should be compatible with the BabelNet model.
        - The ontology should be implemented in OWL.
        - The ontology should be available on-line in different serializations, at least RDF/XML, TTL and HTML.
    - **Functional requirements**: the functional requirements for the EasyTV ontology are being gathered in an online spreadsheet which current content is depicted in Figure 8. It is worth noting that these requirements might evolve along the project lifecycle, therefore they should not be considered as a fixed list of requirements.

4.2.2. EasyTV model

During the development of an ontology intended to be published on-line one of the main decision to be taken is the URI design of the ontology and its elements. For the case of the EasyTV ontology a permanent URI has been selected in order to decouple the ontology physical publication from the identifier of the ontology from which it should be recovered online. More precisely, the “w3id”[35] initiative for permanent URIs has been chosen to identify the ontology URI.

Another important decision is the naming convention to identify ontology elements as classes, object properties, data type properties and instances. In this case the adapted CamelCase in the following way:

- Classes and instances identifiers start with capital letter in the first letter for all the words involved in the term. E.g: VideoFrame
- Object and datatype properties identifiers start with lowercase for the first word and capital

[35] https://w3id.org/

letter for the first letter of the rest of words. E.g: signedForm

Considering the definition of the ontology URI and the notation convention for the ontology elements, the EasyTV ontology and its elements URIs will be form as in the following examples:

- EasyTV ontology URI: https://w3id.org/def/easytv
- example of class URI: https://w3id.org/def/easytv#VideoFrame
- example of property URI: https://w3id.org/def/easytv#signForm

One of the non-functional requirements defined for this ontology is the need of being aligned with the semantic model used in the multilingual knowledge base BabelNet. In addition, reusing ontological resources is one of the best practices in ontological engineering as it maximizes interoperability between models and data annotated with such models while reducing cost and time during the ontology development. For this reason, a number of models for representing linguistic information by using semantic web technologies have been reused, more precisely, those presented in Section 3 as it can be observed in the "Referenced Ontologies" box in Figure 11. More precisely, Table 1 shows the relation of ontologies reused URI and the prefixes used to refer to them along the document.

Table 1. Reused ontology prefixes and namespaces

Prefix	Name	Namespace
bln	BabelNet model	http://babelnet.org/model/babelnet#
lemon	Lemon	http://lemon-model.net/lemon#
lexinfo	Lex Info	http://www.lexinfo.net/ontology/2.0/lexinfo#
owl	OWL	http://www.w3.org/2002/07/owl#
rdfs	RDF(S)	http://www.w3.org/2000/01/rdf-schema#
skos	SKOS	http://www.w3.org/2004/02/skos/core#
vartrans	Variation and translation module of OntoLex	http://www.w3.org/ns/lemon/vartrans

The current conceptual model of the EasyTV ontology is depicted in Figure 11. In such figure the following graphical conventions are used:

Coloured rectangles are used to denote classes created in the EasyTV ontology while white rectangles denote reused classes. For all the entities (classes, object properties and datatype properties), it is indicated in which ontology they are defined by the prefix included before their identifier.

Arrows are used to represent properties between classes and to represent some rdf, rdfs and owl constructs, more precisely:

- Plain arrows with white triangles represent the rdfs:subClassOf relation between two classes. The origin of the arrow is the class to be declared as subclass of the class at the destination

of the arrow.

- Plain arrows between two classes indicate that the object property has declared as domain the class in the origin and as range the class in the destination of the arrow. The identifier of the object property is indicated within the arrow.
- Dashed labelled arrows between two classes indicate that the object property can be instantiated between the classes in the origin and the destination of the arrow. The identifier of the object property is indicated within the arrow.
- Dashed arrows with identifiers between stereotype signs (i.e., “<< >>”) refer to OWL constructs that are applied to some ontology elements, that is, they can be applied to classes or properties depending on the OWL construct being used.
- Dashed arrows with no identifier are used to represent the rdf:type relation, indicating that the element in the origin of the arrow is an instance of the class in the destination of the arrow.

Datatype properties are denoted by rectangles attached to the classes, in an UML-oriented way. Dashed boxes represent datatype properties that can be applied to the class it is attached to while plain boxes represent that the domain of the datatype property is declared to be the class attached.

Individuals are denoted by rectangles in which the identifier is underlined.

Literals are denoted by rectangles in which the value is included between quotation marks.

The representation of additional property axioms (functional, inverse functional, transitive, and symmetric) that are being used in the diagram are shown in the overview ontology legends.

In addition, each ontology overview picture includes a legend to remind the reader the graphics meaning.

As it can be observed in Figure 11 main contributions from the EasyTV ontology are oriented to the inclusion and link of concepts representing Sign Language videos with the linguistic information, mainly reused from existing ontologies. One of the new concepts defined in the EasyTV ontology is etv:Video, created to represent the Sign Language videos generated from the crowdsourcing platform. Every instance of this concept should have the language in which the video is recorded and at least 1 URL indicating the location of the video by means of the attributes lemon:language and etv:url respectively. The videos could be decomposed in frames. In this case, each frame would be represented as an instance of etv:VideoFrame and would be linked to the whole video by means of the property etv:isComponentOf.

As a Sign Language expression, which can be a sentence or phrase, might follow a different word order than the written or oral version of the same sentence, two representation of the constructed sentence are needed, namely one for the Sign Language and another one for the corresponding written language version. For example, the sentence “The three elephants” might be represented in Sign Language as the sequence of the sign for “elephant the three”. The representation of such are modelled by means of the concepts etv:SingedLinguisticExpression and etv:WrittenLinguisticExpression respectively. Both concepts are subclasses, that is, specialisations, of the etv:LinguisticExpression concept. This new concept has been created as the lemon:LexicalEntry concept cannot be reused for this purpose as the sentences intended to be annotated in the EasyTV crowdsourcing platform do not represent an entry in a lexicon. Lemon has been thought for representing entries that have a particular meaning that could be represented by an ontology concept. Therefore, Lemon does not cover the scope of the EasyTV video annotation task because the wording to be annotated could represent phrases or sentences that do not have a specific sense defined by an ontology concept.

It should be mention that lexicons are represented by the class lemon:Lexicon and each lexicon addressed only one language, indicated by the attribute lemon:language. A lexicon is linked to all its lexical entries by means of the relationship lemon:entry.

According to the lemon model, the lexical entries can be realised in different ways from a grammatical point of view. More precisely the lemon model states that a form represents one grammatical realization of a lexical entry. This concept is modelled by the class lemon:Form. Individuals of lemon:LexicalEntry are linked to their forms by means of the property lemon:lexicalForm which is further specialized as lemon:canonicalForm , lemon:abstractForm and lemon:otherForm. Each form can have one or more written representations. In order to adapt this modelling to the EasyTV use case the concept lemon:Form has been extended by adding as subclass the concept etv:SignedForm. In this case, lexical entries are linked to etv:SignedForm by means of the property etv:signedForm which is a sub-property of the lemon:lexicalForm. The signs are realized by videos or video frames and it is indicated by means of the property etv:signedRep.

The rest of the model has been built by merging the necessary modules from existing ontologies to cover the EasyTV requirements to annotate videos with linguistic information in a way compatible with BabelNet model. For the latter reason (compatibility with BabelNet), the original lemon model has been selected for representing lexical entries and senses rather than its reviewed version, lemon-ontolex.[36] Following this model, each lexical entry is linked to one or more sense, represented by the concept lemon:LexicalSense, by the relation lemon:sense. The lexical sense is a bridge term used to unambiguously relate the different meanings of a lexical entry with each ontological concept (represented by the concept skos:Concept) to which they would be linked to by means of the property lemon:reference.

It is worth noting that there are two mechanisms considered in the current model for representing translations. The relation lexinfo:translation (established between senses) which takes into account the sense of the lexical entries being translated and the relation vartrans:translatableAs (established between lexical entries). The lexinfo:translation will be exploited from BabelNet dataset to navigate through senses in different languages while the vartrans:translatableAs would be used, at least in the first iterations, as anchor between lexical entries appearing in the videos being annotated and BabelNet entries.

Being able to annotate morphosyntactic information for each form or lexical entry is an important requirement anticipated in this model. Such annotation will help in refining the search for videos for the EasyTV crowdsourcing platform in order to find the most accurate translation. In this sense, it would be needed to annotate at least: the gender, number, person of an entry and whether it is a negation. Having this future use in mind, the concepts lexinfo:MorphosyntacticProperty, lexinfo:Gender, lexinfo:Negative, lexinfo:Number and lexinfo:Person have been reused as well as the corresponding properties lexinfo:morphosyntacticProperty, lexinfo:gender, lexinfo:negative, lexinfo:number and lexinfo:person.

[36] https://www.w3.org/2016/05/ontolex/

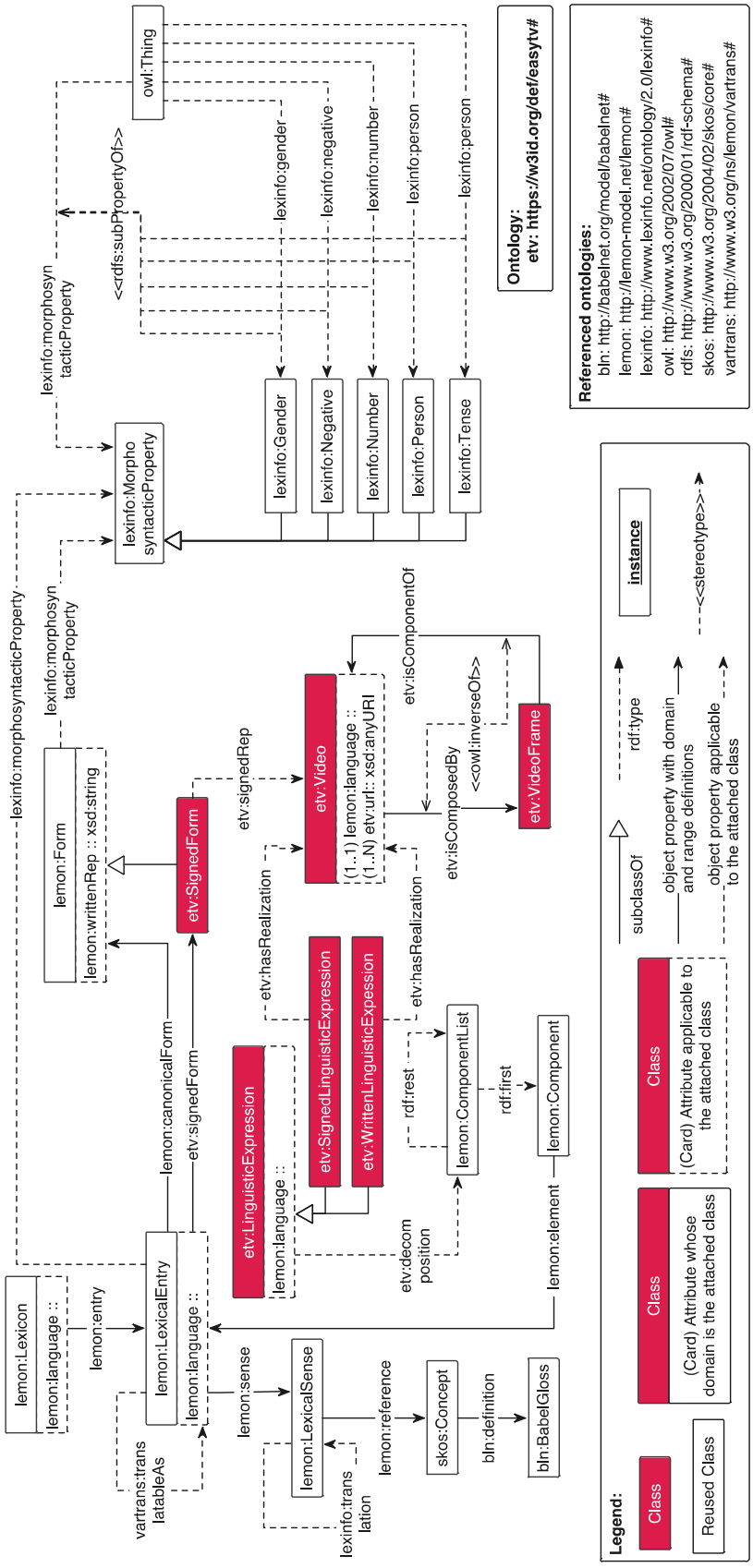

Figure 11. Conceptual model of the EasyTV ontology

Finally, in some cases, it might be needed to decomposed full linguistic expression into smaller units. For this reason, the decomposition modelling provided by the lemon model has been adapted into the EasyTV ontology. This modelling includes the concept etv:LinguisticExpression which is linked to a concept call lemon:ComponentList, which is used to represent lists of elements, by means of the property etv:decomposition. The lemon:ComponentList instances would point to a given element (represented by the concept lemon:Component) by means of the property rdf:first and to the rest of the list (represented by the concept lemon:ComponentList) by means of the property rdf:rest. It is worth noting that the property etv:decomposition has been created instead of reusing lemon:decomposition as lemon:decomposition has domain lemon:LexicalEntry and the linguistic expressions intended to be annotated do no match the lemon:LexicalEntry definition.

In order to show an example of how the ontology is intended to be used to semantically annotate Sign Language videos provided by the crowdsourcing platform an excerpt of an RDF graph is provided in Figure 12. The example provided represents the RDF annotations that should be generated to annotate a given video representing the sentence “The three elephants” which is composed of the sequence of sings of “elephant”, “the” and “three” in that order. It is worth noting that the sign representing “elephant” is linked to the lexical entry elephant in English provided by BabelNet as it can be observed by the prefix “bn” in the instance bn:elephant_n_EN. Such translation link is made by means of the propery vartrans:translatableAs In addition, BabelNet provides the translation of such concept to the Greek sense bn:ελέφαντα_EL/s00030314n.

It should be mentioned that there are ontological requirements defined that should be represented in the ontology and therefore in the data annotations, as for example, the order of the frames within a given video.

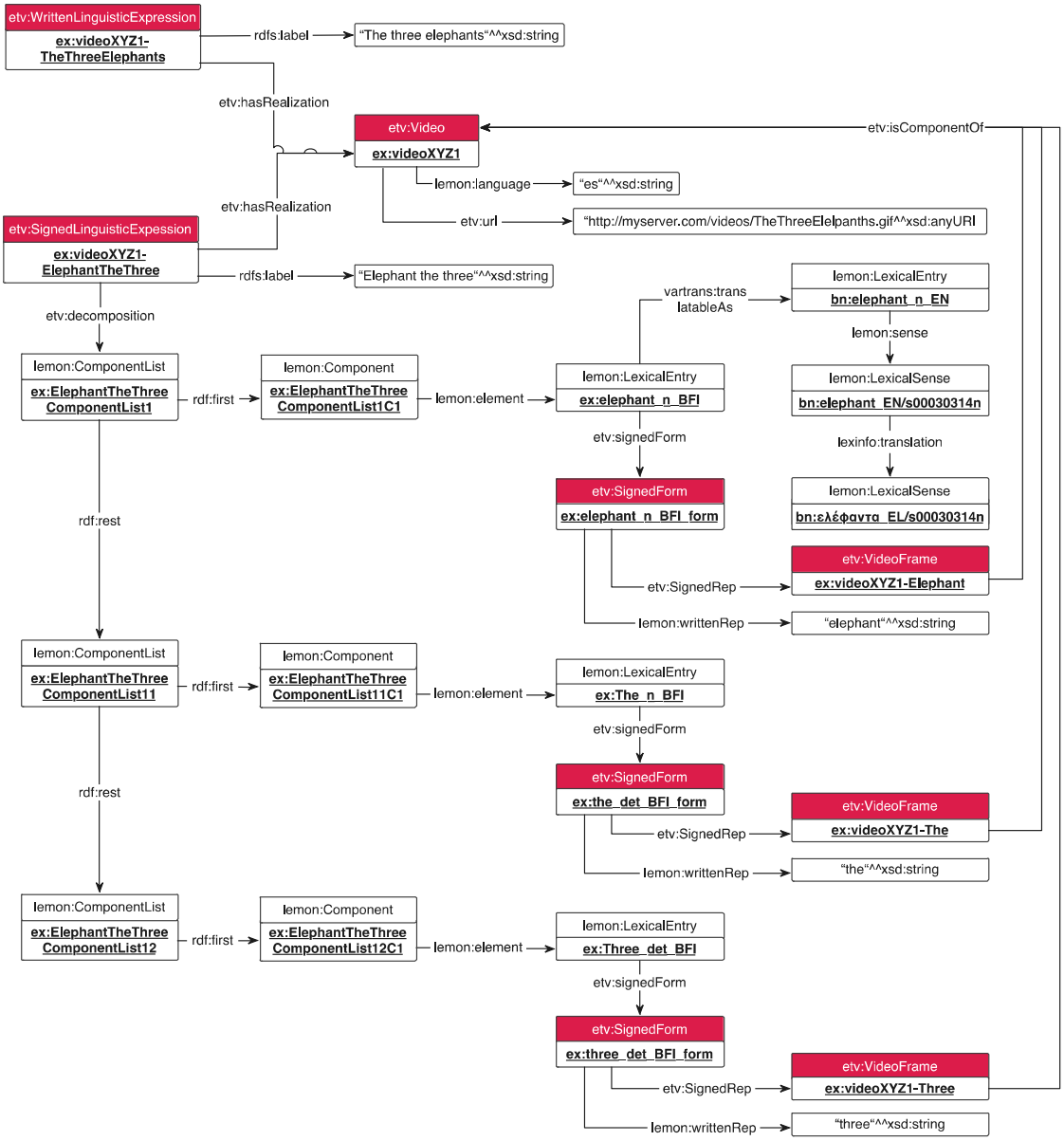

Figure 12. Graphical example of RDF triples

5. NLP TECHNOLOGIES

5.1. Introduction

Task 3.2 *Enrichment of a multilingual ontology with a set of recognizable signs and translation into different languages* needs of signs in different languages associated with their meaning in natural language in order to enrich the ontology through the proposed extension of the model. This signs are received through a video representing a complete sentence or expression in natural language. Sing language videos are segmented in frames with their particular meaning in the sentence or expression (text segment).

The ontology enrichment process has to link a sign and its text segment with the suitable class in the ontology that represent the same meaning of the sign. However, the text segment has not enough information in the process to find a suitable match. Information about the syntactic value of the words, their non-derivative form or how the sentence is structured is needed. This information can be retrieved by Natural Language Processing (NLP) tools and libraries for each specific language. The languages involved in the project are Spanish, Italian, Catalan and Greek. Moreover, English has been also included.

5.2. Requirements

The NLP tools and libraries used for each language should cover the following tasks:

Tokenization: Given a character sequence, tokenization is the task of chopping it up into pieces, called tokens. Those tokens represent the minimum unit of information in a document. Tokens are often loosely referred to as words, but they are not always the same. For instance, symbols (“500$”), abbreviations (“he’s” or “i.e.,”) and specific sequence of characters (plate numbers or IP addresses) can be represented as a unique token or several ones.

Lemmatization: The goal of the lemmatization task is to reduce inflectional forms of a word a word to a common base form, which is typically the one that is represented in dictionaries. For instance, the words “am”, ”is” and “are” derivative forms of the verb “be”. Also, it is used for normalization of words in plural to their singular form (e.g., “hypotheses” to “hypothesis”). In other languages such as Spanish, the lemmatization of words also includes the gender (e.g., “*dura* (hard)” to “*duro* (hard)”).

Part of Speech (POS) tagger: Given a sentence, determine the category to which a word is assigned in accordance with its syntactic functions (noun, adjective, adverb, verb, etc.). Many words, especially common ones, can serve as multiple parts of speech. For example, "book" can be a noun ("read a book") or verb ("to book a room"). Commonly, libraries include the lemmatization task inside the Part-of-Speech tagger.

Named Entity Recognition (NER): Find each mention of a named entity (Real-world concept denoted with a referent term or proper name) in the text and label its type. Default types are the names of persons, organizations, locations, expressions of times, quantities, monetary values and percentages.

Programming language: For the context of the project, the libraries used must be developed in JAVA language or which results can be easily integrated in a JAVA project.

5.3. NLP Tools and Libraries

Different open NLP tools, libraries and products have been studied in order to fix the requirements explained before for each language. The current state of the art is full of private and public libraries for specific languages and specific purposes. For instance, GitHub contains more than 25K repositories for NLP tasks.

The study has been focused over the most common and well-known libraries used in the academia. Finally, four libraries have been selected to cover one or more languages. These libraries will be included in the EasyTV Annotator (see Section 7 to process natural language sentences to contribute in the enrichment process of the ontology. Moreover, an API library has been used to access the data stored in the BabelNet Dataset. The libraries and their characteristics are explained in this section.

5.3.1. CoreNLP

Standford CoreNLP [9] is a JAVA library that provides a set of natural language tools. It can give the base forms of words (lemmas), their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, mark up the structure of sentences in terms of phrases and syntactic dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract particular or open-class relations between entity mentions, etc.

The library covers languages such as English, Arabic, Chinese, French, German and Spanish. After a study of the library, it has been found that the library will be used only for English in the context of the project, as their models have a very high quality.

Stanford CoreNLP[37] is licensed under the GNU General Public License (Full GP), which allows many free uses. For distributors of proprietary software, CoreNLP is also available from Stanford under a commercial licensing

5.3.2. IxaPipes

IXApipes[1] is a modular set of Natural Language Processing tools (or pipes[38]) which provide access to NLP technology for several languages. The IXA pipes architecture concept is oriented as Linux commands in which each of them has a specific task and passes the information to the next through pipes. So processes such as tokenization, part of speech tagging, named entity recognition or sentence parsing are different libraries developed separately, including modules different languages. There are modules for English, Spanish and Basque in each library. Also, there are modules in the part of speech tagging and in the named entity recognition for French, Italian, Galician and German.

The tools are developed by the IXA NLP Group of the University of the Basque Country. The libraries are developed in JAVA language by the IXA NLP Group of the University of the Basque Country and are distributed under the Apache License 2.0 (APL 2.0).

A preliminary study has found that the part of speech tagger and the lemmatizer (which is included in the part of speech tagger) have better results than CoreNLP for Spanish and Italian. So, this library is used to cover both languages.

[37] https://stanfordnlp.github.io/CoreNLP/

[38] http://ixa2.si.ehu.es/ixa-pipes/

5.3.3. **TreeTagger**

TreeTagger [15] is a tool for annotating text with part of speech and lemma information. It was developed in the TC project at the Institute for Computational Linguistics of the University of Stuttgart.[39] TreeTagger is available for multiple languages including English, Spanish, Italian and Greek and is adaptable to other languages if a lexicon and a manually tagged training corpus are available. It is a library developed to work in Shell command or CMD using some scripts in Perl as a stand-alone tool. However, could be easily integrated with other libraries. This library has been used to cover the Greek language in the tasks of part of speech tagger and lemmatizer in combination with CoreNLP.

5.3.4. **Freeling**

FreeLing [13] is a C++ library[40] providing language analysis functionalities (morphological analysis, named entity detection, part of speech tagging, parsing, word sense disambiguation, etc.) for a variety of languages including English, Spanish, Catalan and Italian, among others). Also, it includes an API for JAVA. FreeLing is an open source language analysis tool suite released under the Affero GNU General Public License of the Free Software Foundation. FreeLing project was created and maintained at TALP Research Center, in Universitat Politècnica de Catalunya.

A preliminary study has shown that this library is the only mature project which covers Catalan language in most of the NLP tasks.

5.3.5. **BabelNet API**

BabelNet offers different APIs[41] to access the data of their graph through HTTP or JAVA. Moreover, there is a SPARQL Endpoint which allows access and query the RDF data of the graph through SPARQL queries. However, both APIs offer an easier interface to search lexical entries, synsets and senses through the graph than SPARQL queries. The API queries offer the possibility to specify features of the word directly, such as the written form, the part of speech tag, the language or its lemmatized word. Thus, the quality of the results of the BabelNet APIs to retrieve the best lexical entry in the graph that represents the word or term in natural language, depend on the information retrieved from the NLP libraries.

All the APIs need from a BabelNet API key that it is requested in the web page. The API key provides 1000 Babelcoins per day that represents 1000 queries through any API. It is possible to increase the number of tokens to 50.000 for research purposes. As all the libraries used in the project are in JAVA, the BabelNet JAVA API has been selected.

[39] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

[40] http://nlp.lsi.upc.edu/freeling/index.php/node/1

[41] https://babelnet.org/guide

6. VIDEO, TEXT AND SIGN LANGUAGE REPRESENTATION

For the capturing of Sign Language, we use the EasyTV capturing module, which has been developed in Task 3.1. The module is responsible for acquiring, i.e., capturing and storing, video frame sequences from a depth sensor and extracting information necessary for the construction of motion data, which is used for the realistic avatar playback (in Task 2.4). The capturing process takes place in a room with good lighting conditions and involves a signer speaking Sign Language in front of the depth sensor. Moreover, an admin that controls the capturing software gives instructions to the signer for proper positioning, and marks the start and the end of the capturing process while inspecting the validity of the recordings. The capturing procedure is show below in Figure 13.

In order to produce accurate and noiseless motion data, a series of post-processing steps need to be applied after the image acquisition phase. More specifically, in its current development stage, the EasyTV capturing module uses Microsoft Kinect v2.0 sensor to capture high quality RGB and depth sequences. Both types of imaging data are important, with the former (i.e., RGB) being used by the keypoint detection algorithms while the latter (i.e., depth) being necessary for the generation of 3D data. The sequences of the recorded frames are merged into a single video to be used in the annotation phase.

Figure 13: Testing a multi-view setup for sign capturing.

The video annotation phase comes after the image acquisition process and concerns the assignment of text (i.e., subtitles) that corresponds to each of the recorded signs and unveils their meaning. In case the recorded video contains more than one signs, the annotation process must occur alongside a video trimming process in which the user breaks up the video into proper segments containing the individual signs.

After the completion of the annotation phase, the segmented videos are fed as input into the motion analysis algorithms for the extraction of skeletal data. Skeletal data consist of 3D joints, i.e., keypoints, corresponding to the important visual content of each frame. Due to the nature of Sign Language, we detect 2D keypoints on the hands, face, and body of the signer. Also considering that the detectors currently used in the EasyTV capturing technology are deep networks trained on extracting 2D-keypoints from RGB images, we also need to infer the 3rd dimension for the motion

data, that is, depth. For that reason, an additional processing phase concerns the generation of 3D keypoints by exploiting the depth images captured by the sensor.

The final output of the EasyTV capturing module is a set of files containing both 3D motion data and user annotation information for the recorded signs. These files are subsequently uploaded to the crowdsourcing platform (Task 5.2) as a response to a requested task, and then stored into repositories. The RGB videos recorded can also be exported in order to assist the moderator of the platform in the validation step of the crowdsourcing process. All exported files meet the requirements of EasyTV services and modules, such as the multilingual ontology and the realistic 3D avatar. For a more detailed analysis of the EasyTV capturing technology, refer to D3.1.1 “Sign Language capturing technology preliminary version”.

Before storing the data, the EasyTV crowdsourcing platform has to also obtain annotations given by the multilingual ontology module. These annotations refer to ontology concepts that make it possible to transition from one Sign Language to another. In order to communicate with the multilingual ontology, a communication protocol has to be established. This protocol has to include an invocation procedure and an exchange file format for transferring information. The crowdsourcing platform will invoke a remote procedure or web service and pass a file containing all necessary information as arguments to the multilingual ontology module. Such information might include the whole phrase captured, individual words and video segments corresponding to signs. The multilingual ontology module will respond by returning a file containing the concepts corresponding to the given signs.

In this preliminary version of the deliverable, an initial exchange format for the communication with the multilingual ontology is presented(see Figure 14). More specifically, the communication is performed using a JSON format where the details of each recorded video are given, combined with the annotations provided by the responsible user (e.g., admin or signer) for each sign. Information about the video segments includes the start and the end of each segment, the order in which each sign occurs in the given phrase and the language in which the signer performed the signs.

```
{
    "video": {
        "url": "http://.....",
        "nls": "the tree elephants",
        "sls": "elephant the three" ,
        "duration": "00:50" ,
        "language": "es" ,
                "segmnents": [  {
                "order": "1",
                "start": "00:00",
                "end" : "00:30",
                "content" : "elephant" ,
                }
                {
                "order": "2",
                "start": "00:31",
                "end" : "00:40",
                "content" : "the" ,
                }
                {
                "order": "3",
                "start": "00:41",
                "end" : "00:50",
                "content" : "three" ,
                }]
        }
}
```

Figure 14: An example of a JSON format for describing Sign Language content.

7. EASYTV ANNOTATOR

7.1. Introduction

The easyTV Annotator is a JAVA library developed in the context of the EasyTV European project. The purpose of the library is to enrich the multilingual ontology BabelNet with Sign Language information. For this purpose, the library receives from the crowdsourcing platform a Sign Language video with its natural language transcription sentence and its language. The Sign Language video is segmented in pieces of information representing pieces of the natural language transcription sentence. The library processes the natural language sentence identifying its terms and words. Afterwards, the library finds a suitable representation of these terms or words in the multilingual ontology and associates the video frames to such representations. The library uses different natural language libraries for each specific language. Due to the context of the project and the partners involved in this task, the first version of the easyTV-annotator covers: English, Spanish and Greek.

The public repo is available in the following link: https://github.com/oeg-upm/easytv-annotator

7.2. Architecture

Figure 15 shows the architecture of the **EasyTV Annotator.** The architecture is divided in 5 modules explained in this section.

EasyTV Interface. This module receives the Sign Language information that consists on the sentence and the segmented video and the required language. The module is in charge to manage: the natural language information of the sentence, the video frames, and the information received from the others modules. The module also contains the EasyTV ontology to access and enrich the EasyTV Dataset.

NLP Interface. This module groups all the libraries used in the project for NLP tasks for the different languages with a common interface. These included libraries are coreNLP, Ixapipes, Freeling and TreeTagger, presented in Section 5. All these libraries perform the same tasks: they receive a natural language sentence and they do: tokenization, part of speech tagging, lemmatization, and named entity recognition over it. The results are returned to the EasyTV interface module.

In order to work with all the different libraries, a new implementation of the concepts *Sentence* and *Token* has been created to store the information retrieved from the libraries. The JAVA classes are called ESentence and EToken.

BabelNet Interface. This module contains the methods of the JAVA API to access BabelNet Dataset. BabelNet API methods can retrieve synsets of the ontology specifying language, part of speech of the word and the source of the sysnset. Moreover, the method can retrieve the most relevant sense of a synset for a specific language.

The BabelNet Interface module tries to find a suitable class in the ontology for each of the words (EToken) that composes the sentence, using the information retrieved by the NLP interface module. The result of the BabelNet Interface module for each word is the BabelNet synset and the preferred BabelNet Sense, both stored in their correspondent EToken.

BabelNet Dataset. The BabelNet Dataset is a multilingual encyclopedic dictionary and a semantic network that contains more than 16 millions of multilingual entries stored in RDF triples. BabelNet covers more than 284 languages and integrates information from different sources such as WordNet, Wikipedia or Wikidata.

This module does not belong to the easyTV-annotator library. However, it is present in the architecture because it is used through the communication service of the BabelNet API.

EasyTV Dataset. This module is the SPARQL endpoint where the results of the easTV-annotator library are stored. Each dataset entry contains the sign language information that will enrich the multilingual ontology BabelNet, storing Babelnet synset identified for each word, its language, its part of speech, and the video frame that involves its representation.

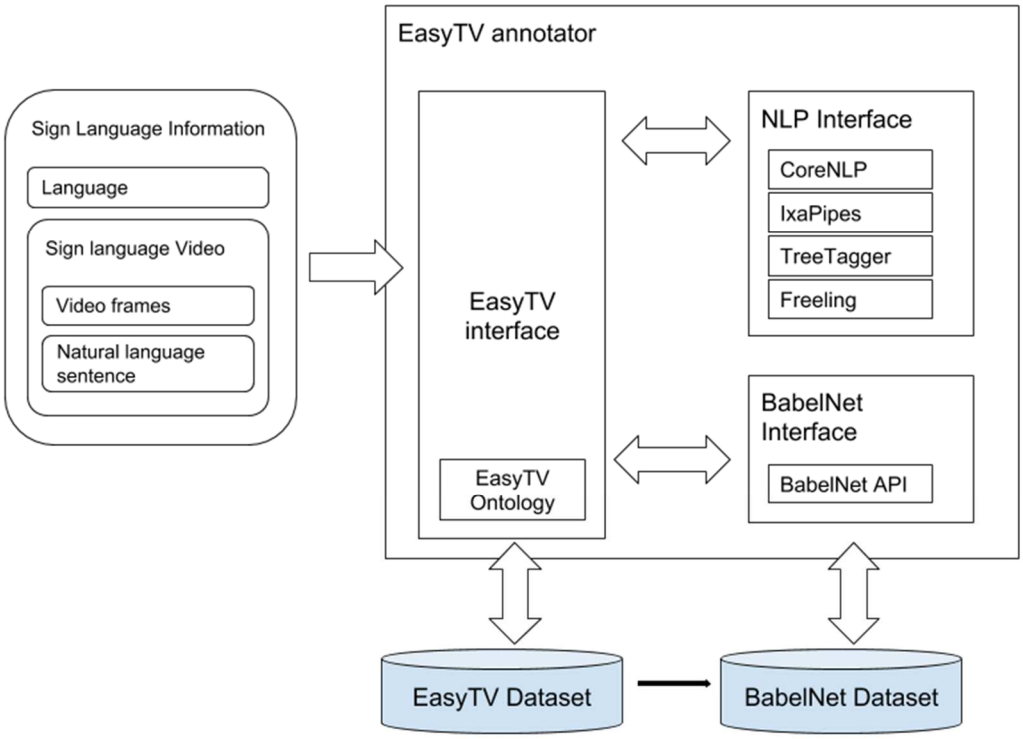

Figure 15. EasyTV Annotator architecture

7.3. Execution

The EasyTV interface module receives a Sign Language video and its language. As Figure 16 shows, the Sign Language video is composed by video frames and a natural language sentence that transcribes the information of the video. This sentence is also composed by tokens associated with specific frames of the video that represents the same piece of information.

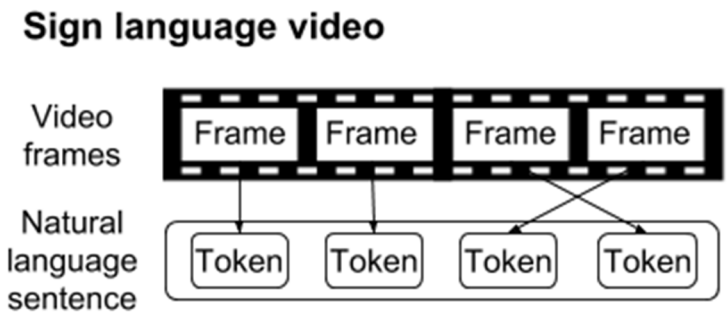

Figure 16. Sign Language video

The EasyTV interface sends the natural language sentence to the NLP interface module to execute the corresponding NLP library for the specific language. This module executes tasks of tokenization, part of speech tagging, lemmatization and named entity recognition with a specific library and creates the internal sentence structure (ESentence) with specific tokens (ETokens) that contain the information needed for the future processes. Figure 17 shows the structure of the ETokens in the ESentence. Tokenization in each library is made to create their proper tokens and finally they are matched with ones received in natural language sentence to create ETokens. The NLP interface module provides the information about the word, the lemma, the part of speech and if the token is a named entity (which commonly involves more than one token).

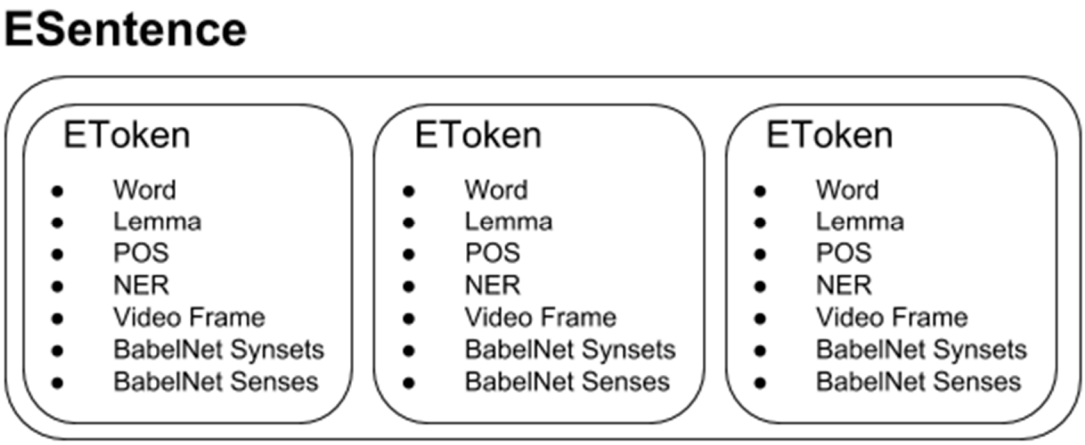

Figure 17. Structure of the ESentence and ETokens

Then, the ESentence is sent to the BabelNet Interface module to retrieve the best candidate class in the ontology for the specific word. The BabelNet Interface uses the provided JAVA Babelnet API using the lemma, the part of speech and the language to retrieve the most accurate synset that represent the word. For disambiguation purposes, the module stores up to three synsets, and for each synset, the best sense for the language is captured.

Finally, the ETokens that contains a video frame and a synset captured are introduced in the EasyTV dataset. New entries are created according to the proposed model in Section 4 linking the video frame and the BabelNet synset.

8. CONCLUSIONS

Along this document the methodology, infrastructure and current version of the EasyTV ontology have been detailed. A review of existing ontologies for linguistic information and an example about how to use the EasyTV ontology to annotate Sign Language videos have also been provided. The current ontology is based on a core module in which sub-models from SKOS, BabelNet, lemon and LexInfo ontologies are reused.

The EasyTV ontology currently published represents an intermediate version that will evolve in the coming months. It covers partially the requirements defined for the model. In addition, the requirements defined for the ontology might evolve according to the crowdsourcing platform development. In this sense, regarding future lines of work, the ontology will be validated according to the needs for the crowdsourcing platforms in terms of annotation of videos and the information needed to retrieve accurate translation of videos (from those annotated in the platform). All the resources produced during the ontology development are available online, and will be updated over time, so they represent the most current view of the ontology to anyone interested.

In addition, this deliverable has described different existing technologies from those analysed, to be reused during the project. More precisely, several semantic models for linguistic information have been presented as well as tools for natural language processing.

As for the ontology-modelling task, several natural language tools have been analysed to be part of the Sign Language annotator that will enrich the multilingual ontology. Natural language processing tools are run in a previous step before implementing the semantic annotation. They are used for tasks such as tokenization, part of speech tagging, lemmatization and named entity recognition with the transcribed information in natural language received from the Sign Language video. The information retrieved by the tools will help in the ontology enrichment process. The tools analysed for the EasyTV project must cover different languages such as English, Spanish, Italian, Catalan and Greek.

Moreover, as BabelNet is the target multilingual ontology that will be enriched with Sign Language information and that represents the base model of the EasyTV ontology, the JAVA API developed by the BabelNet team is used to retrieve suitable classes in the ontology for the words that compose the information. The API has shown that the better results are recovered by lemmatized words in which their part of speech and language is specified. Thus, the quality of the results in this API depends on the results of the NLP tools.

As a result, an annotation library named **easytv-annotator** has been developed to reach the goal of the work package 3.2. The library receives from the word package 3.1 (*Sign Language capturing technology*) Sign Language information in a specific JSON format, containing the language, the Sign Language video, its frames and the transcribed information in natural language.

The library processes the natural language information using the corresponding NLP library to detect tokens, their part of speech, their lemmas and whether there is a named entity. Then, the token words are queried to find a suitable class in BabelNet through the API. Finally, the information of the video frames that compose certain tokens, their information and their BabelNet classes are stored in the EasyTV dataset following the specifications of the EasyTV ontology. In this deliverable, the easyTV-annotator library covers English, Spanish and Greek languages.

9. BIBLIOGRAPHY

[1] Agerri, R., Bermudez, J. and Rigau, G. (2014): "IXA pipeline: Efficient and Ready to Use Multilingual NLP tools", in: Proceedings of the 9th Language Resources and Evaluation Conference (LREC2014), 26-31 May, 2014, Reykjavik, Iceland.

[2] Buitelaar, P., Declerck, T., Frank, A., Racioppa, S., Kiesel, M., Sintek, M., ... & Micelli, V. (2006). LingInfo: Design and applications of a model for the integration of linguistic information in ontologies. In Proceedings of the OntoLex Workshop at Language Resources and valuation Conference, pp. 28-32.

[3] Bosque-Gil, J., Gracia, J., Montiel-Ponsoda, E. y Gómez-Pérez, A. Models to Represent Linguistic Linked Data. Journal of Natual Language Engineering (accepted), Journal of Natural Language Engineering, ISSN: 1351-3249 (Print), 1469-8110 (Online).

[4] Chiarcos, C., Hellmann, S., and Nordhoff, S. (2012). Linking linguistic resources: Examples from the Open Linguistics Working Group, In: Christian Chiarcos, Sebastian Nordhoff and Sebastian Hellmann (eds.), Linked Data in Linguistics. Representing Language Data and Metadata, Springer, Heidelberg, p. 201-216. DOI: https://doi.org/10.1007/978-3-642-28249-2_19

[5] Cimiano, P., Buitelaar, P., McCrae, J., & Sintek, M. (2011). LexInfo: A declarative model for the lexicon-ontology interface. Web Semantics: Science, Services and Agents on the World Wide Web, 9(1), 29-51.

[6] Cimiano, P., Haase, P., Herold, M., Mantel, M., and Buitelaar, P. (2007). LexOnto: A model for ontology lexicons for ontology-based NLP. In Proceedings of the OntoLex07 Workshop held in conjunction with ISWC'07.

[7] García-Castro, R., Fernández-Izquierdo, A., Heinz, C., Kostelnik, P., Poveda-Villalón, M., and Serena, F. (2017) D2.2. Detailed Specification of the Semantic Model. VICINITY project.

[8] Grüniger, M. and Fox, M. (1995) Methodology for the design and evaluation of ontologies. In Skuce, D. (ed.) IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, pp 6.1-6.10

[9] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (pp. 55-60). DOI: http://dx.doi.org/10.3115/v1/P14-5010

[10] McCrae, J., Aguado-de-Cea, G., Buitelaar, P., Cimiano, P., Declerck, T., Gómez-Pérez, A., Gracia, J., Hollink, L., Montiel-Ponsoda, E., Spohr, D., and Wunner, T. (2012). Interchanging lexical resources on the semantic web. Language Resources and Evaluation, 46(4), 701-719. https://doi.org/10.1007/s10579-012-9182-3

[11] Montiel-Ponsoda, E., de Cea, G. A., Gómez-Pérez, A., & Peters, W. (2011). Enriching ontologies with multilingual information. Natural language engineering, 17(3), 283-309. DOI: https://doi.org/10.1017/S1351324910000082

[12] Navigli R., and Ponzetto, S. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. Artificial Intelligence, 193, Elsevier, 2012, pp. 217-250. DOI: http://doi.org/10.1016/j.artint.2012.07.001

[13] Padró, L. and Stanilovsky, E. (2012) . FreeLing 3.0: Towards Wider Multilinguality. Proceedings of the Language Resources and Evaluation Conference (LREC 2012) ELRA. Istanbul, Turkey. May, 2012

[14] Poveda-Villalón, M., Gómez-Pérez, A., and Suárez-Figueroa, M. C. (2014). OOPS! (OntOlogy Pitfall Scanner!): An on-line tool for ontology evaluation. International Journal on Semantic Web and Information Systems (IJSWIS), 10(2), 7-34. DOI: https://doi.org/10.4018/ijswis.2014040102

[15] Schmid, H. (2013). Probabilistic Part-of-Speech tagging using decision trees. In New methods in language processing (p. 154). https://doi.org/10.1007/BFb0026668

[16] Studer, R., Benjamins, V.R., and Fensel, D.: Knowledge engineering: Principles and methods. Data & Knowledge Engineering 25(1-2) (1998) 161-197

[17] Suárez-Figueroa, M. C., Gómez-Pérez, A., and Fernández-López, M. (2012). The NeOn methodology for ontology engineering. In Ontology engineering in a networked world (pp. 9-34). Springer Berlin Heidelberg. DOI: http://doi.org/10.1007/978-3-642-24794-1

ANNEX 1 Methodological guidelines for ontology development

This section presents the ontology requirements specification, implementation, publication and maintenance processes defined for the EasyTV ontology. The development methodology described in this section has been taken from the European project VICINITY [7] and it is originally based on NeOn methodology [17]. The aim of this section is to define the processes to be carried out during the ontology development. Figure 18 shows an overview of the processes that have to be performed and the resultant products. Each of the four steps represented in Figure 18 will be explained in following subsections.

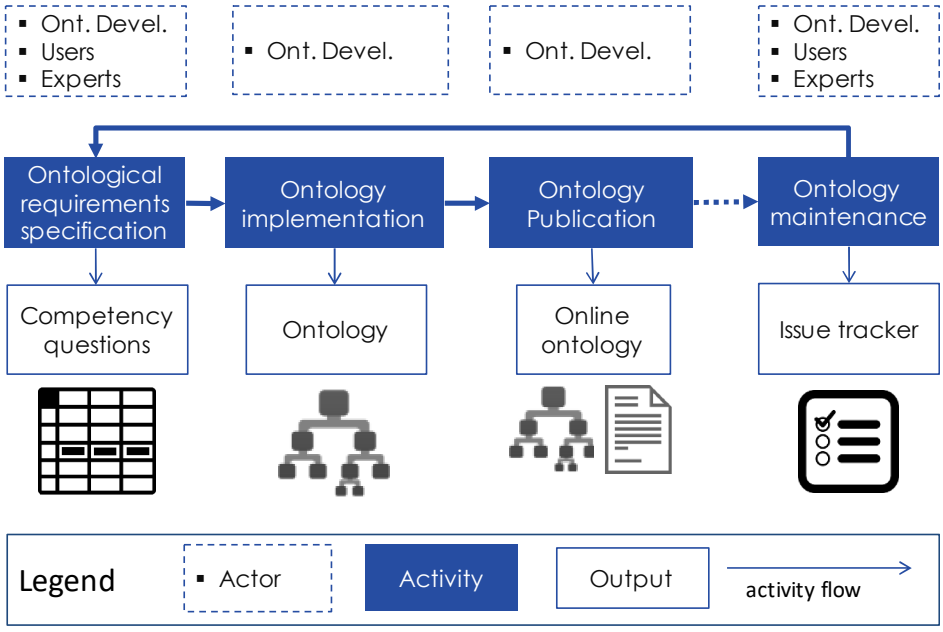

Figure 18. Ontology development process.

1. ONTOLOGICAL REQUIREMENTS SPECIFICATION

The aim of the requirements specification process is to state why the ontology is being built and to identify and define the requirements the ontology should fulfil. In this step, involvement and commitment by domain experts is required to generate the appropriate industry perspective and knowledge.

The activities proposed for the ontology requirement specification process are shown in Figure 19. It should be mentioned that the last step "7. Ontological requirement formalization" does not need to be carried out before the ontology implementation phase necessarily because it does not produce any input needed for implementation. Each activity will be explain in following subsections.

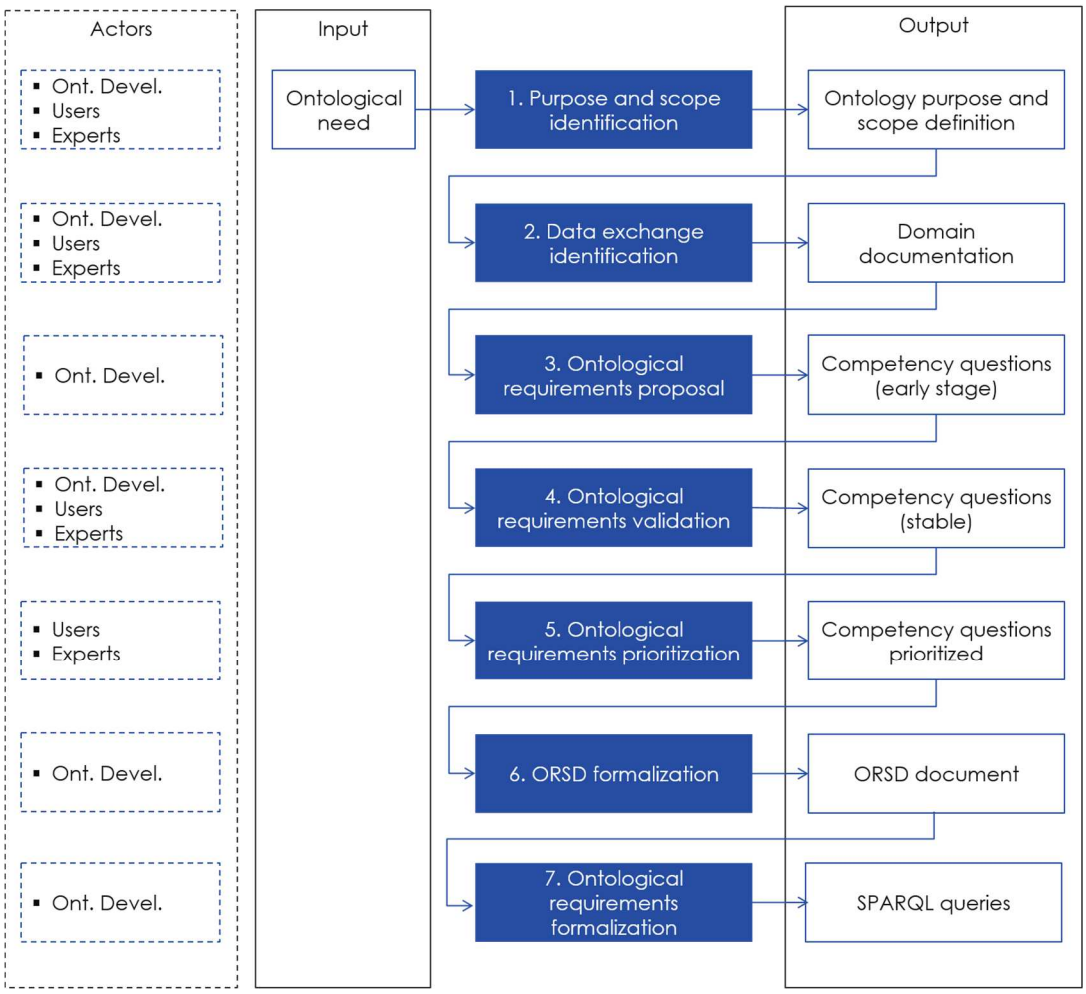

Figure 19. Workflow proposed for ontology requirement specification.

1.1. Purpose and scope identification

The goal of this activity is to define the purpose and scope of the given ontology or ontology module. The ontology development team works in collaboration with users and domain experts to define the purpose and scope of each ontology or module to be developed.

The communication between the domain experts, users and ontology development team could be carried out by means of an online meeting.

1.2. Data exchange identification

The goal of this activity is for the domain experts to provide to the ontology development team the necessary documentation about the domain to be modelled. The documentation to be shared might correspond to:

- Manuals,
- APIs specifications,
- Datasets,
- Standards,
- Formats.

The communication between the domain experts and the development team could be carried out by means of (online) meetings, email or team communication applications.

1.3. Ontological requirements proposal

The ontological requirements could be split into: non-functional and functional requirements according to the NeOn Methodology [17]. On the one hand, the non-functional ontological requirements refer to the general requirements or aspects that the ontology should fulfil, including optionally priorities for each requirement. Some examples of this type of requirements are: “the ontology should be implemented in OWL”, “The ontology should be documented in English and Spanish”, “The ontology should be published on-line following best practices”, etc. On the other hand, the functional requirements refer to the content specific requirements the ontology should fulfil. The requirements are often written in the form of Competency Questions [8] and natural language statements, for example “A video can be decomposed in two or more frames”.

Taking as input the documentation and data provided by domain experts and users, the ontology development team generates a first proposal of ontological requirements written in the form of Competency Questions and natural language statements.

The format used for this proposal follows a tabular approach in which the following fields are included:

- Requirement identifier, which should be unique for each requirement
- Competency question (or natural language statement), which includes:
  - Question
  - Answer
- Provenance information, which includes:
  - Origin of the requirement
- Partners (users or domain experts) related to the definition of the requirement
- Comments about the requirement
- Relation with other requirements
- Priority of the requirements, which can be high, medium or low
- Status of the requirement, which can take the values of “proposed”, “accepted”, “rejected” or “superseded by”
- Sprint where the requirements must be implemented

1.4. Ontological requirements completion and validation

This activity will focus on validating the ontology requirements defined in the previous step. For doing so, domain experts and users in collaboration with the ontology development team would check whether the ontology requirements are correct and complete.

The following criteria can be used in this validation task as stated in [8]:

- A set of requirements is correct if each requirement refers to some features of the ontology to be developed.
- A set of requirements can be considered complete if users and domain experts review the requirements and confirm they are not aware of additional requirements.
- A set of requirements can be considered internally consistent if no conflicts exist between the requirements.

- A set of requirements is verifiable if there is a finite process with a reasonable cost testing whether the final ontology satisfies each requirement.
- Each requirement must be understandable to users and domain experts.
- An ontology requirement is unambiguous if it has only one meaning; that is, if it does not suggest any doubt or misunderstanding.
- A set of requirements is concise if all requirements are relevant and there are no duplicates.
- A set of requirements is realistic if each and every requirement meaning represents a characteristic of the domain.

The communication between the domain experts and the development team could be carried out by means of (online) meetings, email or team communication applications.

1.5. Ontological requirements prioritization

This activity will be performed if there is the need for prioritizing functional requirements. The main implications of this prioritization are the possibility of planning and scheduling the development of the ontology in sprints. This prioritization would be present in the backlog that will drive the ontology development process.

To carry out this prioritization the ontology development team works with the domain experts to identify which requirements need to be fulfilled first. The communication between the domain experts and the ontology development team could be carried out by means of an on-line or in-person interview.

1.6. ORSD formalization

This activity will generate the consolidated document of the requirements. Once the ontology development team has all the information about the requirements, they create the Ontology Requirements Specification Document (ORSD). This specification document stores all the requirements identified and the information associated to them.

1.7. Ontological requirements formalization

This activity will generate the translation of the ontological requirements, written in natural language, into a formal language, into test cases. These tests cases, which are stored in RDF files, should include:

- Identifier of the requirement associated,
- Description of the test case, which includes a link to the ORSD,
- SPARQL query extracted from the competency question,
- Expected result of the query.

The goal of this activity is to create machine-readable test cases. These test cases have SPARQL queries which can be executed over the ontology to verify if the ontology satisfies the ontological requirements identified.

2. ONTOLOGY IMPLEMENTATION

The aim of the ontology implementation process is to build the ontology using a formal language, based on the ontological requirements identified by the domain experts. After defining the first set of requirements, though modification and addition of requirements is allowed during the development,

the ontology implementation phase is carried out through a number of sprints. The ontology developers schedule and plan the ontology development according to the prioritization of the requirements in the ontology requirements specification process. The ontology development team builds the ontology iteratively, implementing only a certain number of requirements in each iteration. The output of each iteration is a new version of the ontology.

Figure 20 shows the activities to carry out during the ontology implementation for each iteration.

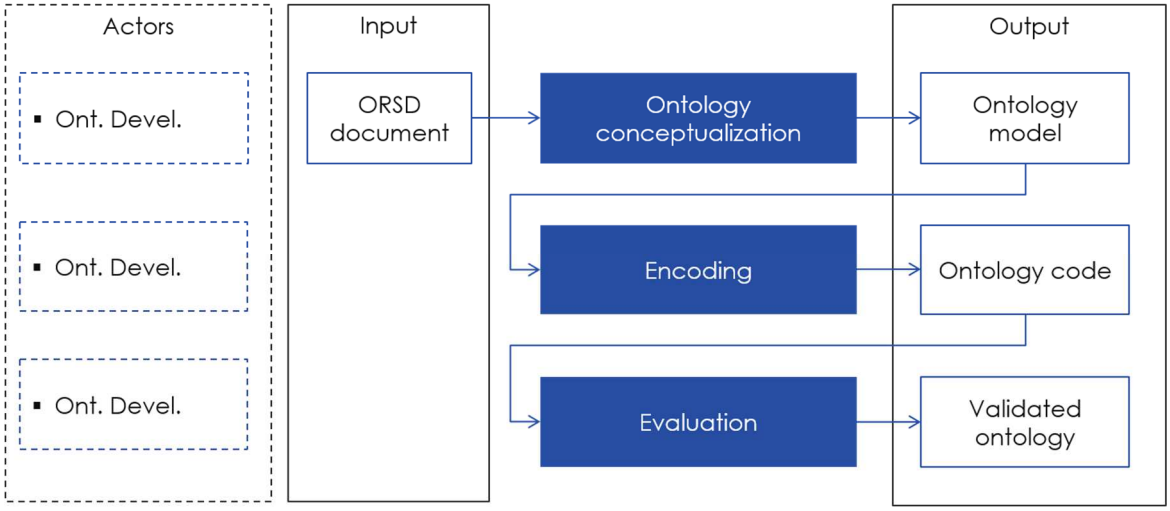

Figure 20. Workflow proposed for ontology implementation.

2.1. Ontology conceptualization

The aim of this activity is to build an ontology model from the ontological requirements identified in the requirements specification process. During the ontology conceptualization, the domain knowledge obtained from the ORSD document is organized and structured into a model by the ontology developers.

2.2. Encoding

During this activity, the ontology development team generates computable models in the OWL language from the ontology model.

The ontology code resultant from this activity includes metadata, such as creator, title, publisher, license and version of the ontology.

It is worth noting that during the development of the EasyTV ontology the following *Ontology Versioning* convention has been adopted:

The versioning identification will be as similar as possible to the conventions used in software development. In this case, each release will follow the pattern *v.major.minor.fix*, where each field follows the rules:

- **major**: The field is updated when the ontology covers the complete domain it intends to model. That is, it is a complete product and covers the final goal of the development.
- **minor**: The field is updated when:
  - all the requirements of a subdomain are covered.

- documentation is added to the ontology.
- **fix**: The field is updated when:
  - typos or bugs are corrected in the ontology.
  - classes, relationships, axioms, individuals or annotations are added, deleted or modified.

In each iteration the minor and fix fields might be changed from zero to several times. When developing ontology networks, it could be the case that each module follows its particular version status. For example, one ontology might be in version v1.0.0 while the another ontology of the network might be in version v0.2.3. That is, the network evolution is not managed as a whole, as it is not a particular product but the virtual composition of many ontologies where each ontology evolves independently.

2.3. Evaluation

Before publishing a release version of the ontology, the developers evaluate the ontology in different aspects:

- The ontology developers guarantee that the ontology does not have syntactic, modelling or semantic errors.
- The ontology developers guarantee that the ontology fulfil the requirements scheduled for the ontology using the test cases generated in the requirements specification process.

3. ONTOLOGY PUBLICATION

The aim of the ontology publication process is to provide an online ontology accessible both as a human-readable documentation and a machine-readable file from its URI. The ontology needs to be evaluated before its publication to guarantee that is ready to be used. Figure 21 shows the steps proposed to carry out the ontology publication activities, which are further described in next subsections.

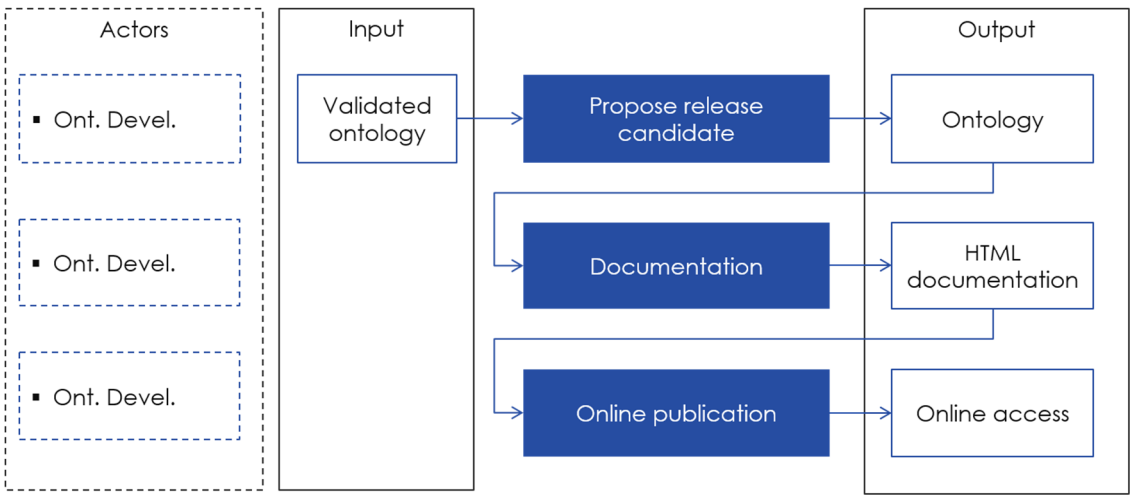

Figure 21. Workflow proposed for ontology publication.

3.1. Propose release candidate

Once the ontology developers have implemented and validated the ontology, they propose a release

version of the ontology to be published on the Web:

- In case the ontology fulfils all the requirements of a given subdomain, the ontology development team generates a release version of the ontology, e.g., the EasyTV ontology for release 0.1 is tagged as “release version v0.1.X”.
- In case the ontology does not implement all the ontological requirements identified, only those scheduled for the iteration, the ontology development team generates a pre-release version of the ontology.

Both release and pre-release versions of the ontologies are evaluated and ready to be used.

3.2. Documentation

Taking as input the ontology generated in the previous activities, the ontology development team in collaboration with the domain experts generates the ontology documentation of the release candidate. This documentation includes:

- An HTML description of the ontology which describes the classes, properties and data properties of the ontology, and the license URI and title being used. The domain experts have to collaborate with the ontology development team to describe the classes and the properties. This description also includes metadata, such as creator, publisher, date of creation, last modification or version number.
- Diagrams which store the graphical representation of the ontology, including taxonomy and class diagrams.

3.3. Online publication

Once the documentation of the ontology has been generated, the ontology is published on the Web. This online ontology is accessible via its namespace URI as a machine-readable file and a human-readable documentation using content negotiation.

4. ONTOLOGY MAINTENANCE

The goal of this activity is to update and add new requirements to the ontology that are not identified in the ORSD or to identify and corrects errors or to schedule a new iteration for ontology development. During the ontology development process, the domain experts can propose new requirements or improvements over the ontology. If these requirements or improvements are approved by the ontology development team, they are added to the ontology.